Novel energy efficient compute architectures on the path to Exascale.

Ioan Corneliu Hadade

August 24, 2012

MSc in High Performance Computing
The University of Edinburgh
Year of Presentation: 2012
Abstract

This project conducts research on the area of energy-efficient compute architectures which are primordial for the progression to Exascale. An evaluation of the Tibidabo ARM cluster is performed with the usage of synthetic benchmarks and real HPC applications in order to assert the performance of ARM-based CPU’s as building blocks for HPC systems.
## Contents

1 **Introduction** 1
   1.1 Aims and Objectives ........................................ 3
   1.2 Report Summary ............................................. 4

2 **Background** 5
   2.1 Exascale Challenge ........................................ 5
      2.1.1 Mont Blanc ............................................. 7
   2.2 Energy Efficient Compute Architectures .................... 9
      2.2.1 ARMed for HPC? ......................................... 9
      2.2.2 CUDA on ARM ........................................... 12
      2.2.3 Project Denver .......................................... 15
      2.2.4 Project Echelon ......................................... 16
      2.2.5 BlueGene/Q ............................................. 19
      2.2.6 Intel Xeon Phi .......................................... 22
      2.2.7 Summary ................................................ 24

3 **Tibidabo** 25
   3.1 Architecture Overview ..................................... 25
   3.2 Software Overview .......................................... 28
   3.3 Power Efficiency ............................................ 29
   3.4 Issues ..................................................... 30
   3.5 Summary .................................................... 30

4 **Benchmarks** 31
   4.1 Evaluation methods .......................................... 31
   4.2 Synthetic Benchmarks ....................................... 32
      4.2.1 HPC Challenge Benchmark ................................ 32
   4.3 Application Benchmarks ..................................... 33
      4.3.1 Jacobi Iterative Solver .................................. 33
      4.3.2 Black Scholes Option Pricing ............................. 34
      4.3.3 Particle Simulation ....................................... 34
   4.4 Summary ..................................................... 35

5 **Results and Analysis** 36
   5.1 Compiling Optimisations .................................... 36
5.2 HPC Challenge Benchmark . . . . . . . . . . . . . . . . . . . . 37
  5.2.1 HPL . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 37
  5.2.2 DGEMM . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 39
  5.2.3 STREAM . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 39
  5.2.4 PTRANS . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 39
  5.2.5 RandomAccess . . . . . . . . . . . . . . . . . . . . . . . . . . 39
  5.2.6 FFT . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 39
  5.2.7 Ping-Pong . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 40
  5.3 HPC Codes . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 40
    5.3.1 Jacobi Iterative Solver . . . . . . . . . . . . . . . . . . . . . 40
    5.3.2 Black Scholes Option Pricing . . . . . . . . . . . . . . . . . 42
    5.3.3 Particle Simulation . . . . . . . . . . . . . . . . . . . . . . . 44
  5.4 Summary . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 47

6 Conclusions 48

7 Further Work 49

8 Project Post-Mortem 50

A Synthetic Benchmarking Results 52
  A.1 Single Node HPL Output . . . . . . . . . . . . . . . . . . . . . . 52
  A.2 Full System HPL Output . . . . . . . . . . . . . . . . . . . . . . 53

B Slurm Job Script 54

C Wilson and Clover QCD Scaling Benchmark 55
List of Tables

5.1 GCC ARM Optimisation Flags for serial MD code ................. 36
5.2 HPL Performance on maximum core count ...................... 38
5.3 DGEMM Benchmark in GFlops/s ................................. 39
5.4 Star Stream Benchmark with vector size 205761 ................. 39
5.5 Single Stream Benchmark with vector size 205761 ............... 39
5.6 PTRANS Benchmark Results ..................................... 39
5.7 GUPs Star and Single Random Access N=524288 ................. 39
5.8 LCG GUPS Star and Single Random Access N=524288 .......... 39
5.9 Star and Single FFT Benchmark with FFT N=131072 .......... 39
5.10 Ping Pong Benchmark (Latency=useconds; Bandwidth=GB/s) .. 40
5.11 Black-Scholes Asian Option Pricing Execution Times in seconds .... 44
5.12 Particle Simulation Execution Time in seconds ................. 46
5.13 Computation vs. Communication in seconds on Tibidabo ....... 47
5.14 Computation vs. Communication in seconds on HeCToR ....... 47
# List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Mont Blanc Roadmap</td>
<td>8</td>
</tr>
<tr>
<td>2.2</td>
<td>Mali-T678 GPU</td>
<td>11</td>
</tr>
<tr>
<td>2.3</td>
<td>Carma Development Kit</td>
<td>13</td>
</tr>
<tr>
<td>2.4</td>
<td>NVIDIA Kepler - Dynamic Parallelism</td>
<td>14</td>
</tr>
<tr>
<td>2.5</td>
<td>NVIDIA Kepler - Hyper-Q</td>
<td>15</td>
</tr>
<tr>
<td>2.6</td>
<td>NVIDIA Echelon Architecture</td>
<td>17</td>
</tr>
<tr>
<td>2.7</td>
<td>Energy cost of data movement</td>
<td>17</td>
</tr>
<tr>
<td>2.8</td>
<td>Flat vs. Hierarchical Memory</td>
<td>18</td>
</tr>
<tr>
<td>2.9</td>
<td>Blue Gene/Q Power BQC architecture</td>
<td>20</td>
</tr>
<tr>
<td>2.10</td>
<td>Intel Xeon Phi Die</td>
<td>22</td>
</tr>
<tr>
<td>2.11</td>
<td>Intel Xeon Phi Card</td>
<td>23</td>
</tr>
<tr>
<td>3.1</td>
<td>NVIDIA Tegra 2 SoC</td>
<td>26</td>
</tr>
<tr>
<td>3.2</td>
<td>Tibidabo Compute Blade</td>
<td>27</td>
</tr>
<tr>
<td>3.3</td>
<td>Tibidabo Prototype Rack</td>
<td>27</td>
</tr>
<tr>
<td>3.4</td>
<td>Tibidabo Software Environment</td>
<td>28</td>
</tr>
<tr>
<td>3.5</td>
<td>Tibidabo Node Power Consumption</td>
<td>29</td>
</tr>
<tr>
<td>5.1</td>
<td>High Performance LINPACK Performance</td>
<td>37</td>
</tr>
<tr>
<td>5.2</td>
<td>HPL Power Consumption</td>
<td>38</td>
</tr>
<tr>
<td>5.3</td>
<td>Tibidabo position in Green Top 500</td>
<td>38</td>
</tr>
<tr>
<td>5.4</td>
<td>2D Jacobi Solver 10000 Iterations on Tibidabo</td>
<td>40</td>
</tr>
<tr>
<td>5.5</td>
<td>2D Jacobi Solver 20000 Iterations on HeCToR</td>
<td>41</td>
</tr>
<tr>
<td>5.6</td>
<td>2D Jacobi Solver Speedup</td>
<td>42</td>
</tr>
<tr>
<td>5.7</td>
<td>Black-Scholes Asian Option Pricing on Tibidabo</td>
<td>43</td>
</tr>
<tr>
<td>5.8</td>
<td>Black-Scholes Asian Option Pricing on HeCToR</td>
<td>43</td>
</tr>
<tr>
<td>5.9</td>
<td>Black-Scholes Asian Option Pricing Speedup</td>
<td>44</td>
</tr>
<tr>
<td>5.10</td>
<td>Particle Simulation Scalability on Tibidabo</td>
<td>45</td>
</tr>
<tr>
<td>5.11</td>
<td>Particle Simulation Scalability on HeCToR</td>
<td>45</td>
</tr>
<tr>
<td>5.12</td>
<td>Particle Simulation Speedup</td>
<td>46</td>
</tr>
<tr>
<td>C.1</td>
<td>QCD Scalability on Tibidabo</td>
<td>55</td>
</tr>
</tbody>
</table>
Acknowledgements

This project would not have been possible without the kind contribution of others. I would like to thank my supervisor James Perry for his constant guidance and advice throughout this project and especially during times when unexpected issues forced a change in trajectory and objectives.

Many thanks go to Dr. Alex Ramirez and Nikola Rajovic from the Barcelona Supercomputing Centre for allowing me to use the Tibidabo cluster and the ever so prompt replies I have received to all my queries. Without your excellent prior work, this project might have not materialised.

I would like to thank Dr. Stephen Booth for providing me with the necessary codes and crash course introduction into Quantum Chromodynamics and EPCC for what has been a truly exceptional learning experience.

And finally, I would like to thank my parents and Melanie and Kevin Smith for their unconditional love and support that I have been blessed with throughout my life.
Chapter 1

Introduction

Today’s Petascale systems are capable of performing in excess of a quadrillion or $10^{15}$ floating point operations per second. The utilisation of such enormous computational power serves as a catalyst for man’s endeavour to analyse, model, comprehend and predict the complex behaviour and interactions of the surrounding physical systems [1]. For instance, scientists are now able to simulate the full operation of a nuclear reactor at temperatures exceeding 100 million degrees centigrade before even laying the first slab of concrete and therefore pre-empting prospective safety hazards or maximum throughput at the initial design phase [2].

Although HPC systems are being used for the simulation of a plethora of other applications spanning a multitude of science fields from bioinformatics to finance, the above example has been mentioned with a purpose. Perhaps ironically, it is believed that a transition to Exascale ($10^{18}$ floating point operations per second) using today’s technology will most likely require that each supercomputer have their own small nuclear reactor in order to sustain its insatiable thirst for power [1]. This is particularly alarming since exaflop computation is imperative if one wishes to understand the fundamental components of the universe or meet a nation’s goals for energy, security and ecological sustainability [3].

The underlying reason for the necessity of such large quantity of energy required for running future supercomputers is based on the technology that current HPC systems rely upon, specifically the usage of commodity components that act as primordial building blocks [4]. The recent transition from Teraflop ($10^{13}$) to Petaflop ($10^{15}$) has been achieved through the migration from single core to multi-core processors[1]. Previous to that, the path from Gigaflop ($10^{10}$) to Teraflop was possible due to the increase in the chip’s clock frequency therefore allowing it to execute a higher number of computations per clock cycle [5].

The issue with both approaches is the compromise they makes between performance and power efficiency. In order to arrive at a sweet spot between the two, one has to pay special attention to the fundamental law governing the power utilisation of CMOS
circuits, namely, the dynamic power equation:

\[ P = ACV^2 F \]

In the above equation, \( P \) represents the power that is consumed, \( A \) the activity factor or the part of the circuit that is switching, \( C \) the capacitance, \( V \) the voltage and \( F \) is the clock frequency [6]. Therefore, a charge and discharge of \( C \) at a frequency \( F \) and voltage \( V \) results in power consumption obtained by computing \( CV^2 F \).

As one can observe, increasing the frequency rate at which a processor operates will inevitably lead to higher power consumption as the dynamic power tends to grow linearly with it. Furthermore, empirical results have demonstrated that reducing the clock speed by a mere 20% can in fact lead to a 50% drop in energy utilisation [4] as it also leads to a significant drop in voltage which in itself is a square problem. As a result, chip makers decreased frequency whilst adding more cores on the same die in order to compensate with the inevitable performance loss.

This approach delivered promising results with multi-core chips registering performance increases in the region of 70% for the same energy intake over single core [7]. However, constantly adding more cores inevitably leads to yet another obstacle. First of all, a larger core count results in a higher density which requires a continuous shrinkage in transistor size. This process is unsustainable as one gets closer to the current limits set by quantum mechanics since there are no known methods of scaling lower than the size of a single atom [8]. Furthermore, in order to expose parallelism, multi-core processors have to constantly keep caches coherent which in turn requires an increased amount of data movement on and off-chip. This in turn leads to an increase in the required on the wire voltage which results in more power utilisation.

To put this further into context, moving data across a 20mm die requires a 3x increase in energy over the power required to compute a 64-bit matrix addition [4]. This figure increases by a factor of ten when accessing data off-chip where a high core count results in an even lower memory bandwidth since this has to be shared across all existing cores on a die [7].

Even the most power efficient microprocessors today can only deliver approximately 450 Mflops per Watt when running Linpack which is one hundred times less compared to the 50 Gflops per Watt target required for exascale and published in a DARPA commissioned report [9]. Extrapolating this number results in approximately 2.2 GigaWatts needed for computing an exaflop without even considering the necessary power needed for cooling which can add an extra 20-100% increase in the power draw [4]. This figure is worrying as a typical nuclear power plant can deliver in the neighbourhood of 2.5 GigaWatts of power and makes the initial analogy of one nuclear power plant per supercomputer less ludicrous as it might have seemed initially.

It is therefore without a doubt that one must look at pursuing different alternatives if reaching exascale compute capabilities is to ever become a tangible reality. One such alternative could be the utilisation of System On-Chip devices as processing elements.
within an HPC system replacing the large power-hungry commodity processors in use today [10]. There seems to be a new found perception that the embedded market might be able to offer the necessary expertise and processing components in tackling the power wall problem. After all, the embedded world have been operating in a stringent power bracket environment for a very long time with the constant focus of squeezing the highest amount of performance at the lowest possible energy intake [11]. Furthermore, the HPC industry is currently witnessing a success story of embedded processors powering Petascale supercomputers.

The IBM Blue Gene series uses custom processors based on the PowerPC 440 architecture which was initially designed for the embedded market [12]. The latest incarnation of the Blue Gene, namely, the Q series, is a supreme leader in the energy efficiency contests occupying the leading positions within the Top 500 Green List by a hefty margin [13].

A similar approach is also currently under way at the Barcelona Supercomputing Centre under the Mont-Blanc project. In a session at the International Supercomputing Conference ’12, Dr. Alex Ramirez presented the centres endeavour in building an ARM-based cluster using commodity embedded processors found in today’s smartphones [14]. The particular advantage of this initiative over the Blue Gene’s case is that of cost. The price of procuring these ARM-based processors is significantly lower since all R&D costs are backed up by a fast growing smartphone market which is predicted to be worth in excess of 300B dollars by the end of 2015 [15]. The project’s ultimate goal is to build an exascale system that utilises up to 30 times less energy compared to current supercomputing architectures through the synergy of leading edge ARM chips and energy efficient accelerators such as the newly released NVIDIA Kepler [16].

The initial objective of this project was to empirically evaluate whether this energy efficient heterogeneous platform based on the marriage of a low-powered ARM chip and an energy efficient GPU accelerator is indeed a suitable option for achieving superior computational performance over reduced power draw using the recently announced NVIDIA Carma development kit [17]. However, due to issues in procurement, this was not at all possible as constant shipping delays crippled the entire endeavour to a halt. As a result, focus has been shifted on benchmarking the capabilities of Barcelona Supercomputing Centre’s first incarnation of the Mont-Blanc architecture roadmap, namely, the Tibidabo cluster based on low-powered NVIDIA Tegra 2 SoC.

1.1 Aims and Objectives

Due to the issues presented above, the project aims and objectives suffered slight alterations to the ones previously described in the project proposal report. This project will now aim to explore the capabilities of the Tibidabo cluster with an emphasis on ARM’s current capabilities as a processing building block for the energy efficient compute architectures required on the road to Exascale.
The evaluation of the cluster shall be done using empirical methods such as:

- Synthetic benchmarking using the HPCC suite.
- Application benchmarking using a set of real scientific application codes such as Quantum Chromodynamics, Monte Carlo and Particle Simulation and Iterative Solvers.

The primary focus of all of the above will be placed in relevance to energy efficiency with comparisons to current HPC architectures being made where necessary and deemed reasonable.

1.2 Report Summary

Chapter 1 aimed to describe the project’s motivation for researching on the area of energy efficient compute architectures by presenting the limitations of current compute architectures in regards to the energy efficiency characteristics that are required for Exascale computing. This section concluded with the project’s aims and goals in which its subsequent objectives were laid out in the context of the identified problem.

Chapter 2 reviews the challenges facing the HPC community in its progression to Exascale and ARM’s prospective usage as a future energy efficient architecture. This is continued with a detailed description and appraisal of current and future energy efficient compute architectures in the pursuit of low-power high performance computing.

Chapter 3 describes the Tibidabo cluster used as the project’s evaluation platform.

Chapter 4 presents the benchmarking methods to be used on the Tibidabo and other systems deemed necessary for comparison reasons.

Chapter 5 contains the results and analysis of the executed benchmarks.

Chapter 6 postulates the conclusions and experiences drawn from trying to complete the project’s aims and goals.

Chapter 7 anticipates the future work that can be carried out in the area of energy efficient compute architectures and which would benefit from this project’s output.

Chapter 8 concludes with a short description of some of the external issues that have been faced throughout this project.
Chapter 2

Background

The aim of this chapter is to present an overview of the literature available on the area of energy efficient high performance computing. A critical review of current and future compute architectures is given with a view on ARM’s position within the HPC ecosystem alongside collaborative initiatives that aim to bring forth a resolution to the power obstacle standing in front of progression towards Exascale computing.

2.1 Exascale Challenge

The Exascale Challenge is known in the supercomputing field as the technological hurdle that one needs to address in order to facilitate the sustainable transition from Petascale to Exascale systems. According to a report commissioned by DARPA [9], for this transition to become a reality, a complete overhaul is needed on the current paradigms and architectures used for Petaflop supercomputers by giving a careful consideration to the following issues:

- Energy and Power
- Memory and Storage
- Concurrency and Locality
- Resiliency, Fault Tolerance and Validation

The HPC community have come to categorise the above into separate distinct challenges each exhibiting individual characteristics [33].

The Energy and Power challenge is determined by the already described issue of current power utilisation levels within Petascale supercomputers and across all commodity components that are used to build them. The DARPA report suggests that reducing the power draw for future Exascale systems will be the most difficult challenge to tackle since it would require an entirely different architecture than the ones currently used. This view is shared across the entire supercomputing community[34][35][36]
with some believing that embedded microprocessors currently used in smartphones may be able to replace the x86 power-hungry CPU’s due to their proven energy efficient and commodity nature [14]. This project aims to establish just that by investigating the performance and power-efficiency of the "world’s first ARM-based cluster", Tibidabo.

**Memory and Storage** is the second challenge illustrated in the DARPA report and deals with the current discrepancies in latency and bandwidth that exist between the CPU and main memory together with the headaches of storing exaflops of data that simulations on future supercomputers might generate [4].

As processors maintained Moore’s Law’s status quo by either increasing frequency or the number of cores on a die, main memory has not experienced such a progressive slope with only an order of magnitude improvement in both bandwidth and latency being achieved over the last decade. Furthermore, not only is memory currently lacking behind in performance, it is also one of the main culprits for high power draws within a system due to the volatility nature of technologies such as DRAM.

Building supercomputers out of millions of low-powered processors whilst still using the current memory architecture will not alleviate the power consumption problem as one would have to increase the number of channels to cope with millions of threads fetching and writing back data [34]. This will inevitably lead to an increase in the power consumed bearing in mind that as previously stated, data movements off-chips are 4x more expensive in terms of energy utilisation and will lead to memory becoming an even worse enemy to power efficiency than seen current CPU’s in Petascale supercomputers.

For these reasons, increased research has been focused on finding a suitable solution for Exascale memory and storage by investigating other alternatives and materials such as chalcogenide glass used in developing phase-changed memory [8] or graphene [4].

The **Concurrency and Locality** challenge is in fact an issue that is currently being experienced in Petascale computing and will no doubt be exacerbated when migrating to Exascale. Scientific applications usually exhibit certain characteristics depending on their domain and the science they try to solve. Such characteristics range from computationally bound kernels where simply throwing more compute resources at the problem results in immediate speed up to communication bound classes where the actual bottleneck lies within the system’s communication pattern, interconnect, algorithm or programming model used [4].

There has been a number of research endeavours to investigate this particular problem such as the CRESTA project [37]. Their research suggests that the accelerated evolution of hardware is leaving the majority of HPC applications behind where codes are unable to scale over 512 cores [38]. This is indeed worrying as the current TOP500 leader, the Sequoia IBM Blue Gene/Q, contains over one million and a half cores. Their solution sees the principle of co-design where hardware is built and configured with the software that’s destined to run on it in mind, a view that is also shared across certain parts of the HPC community [26][39].

6
One of the aims of this project is to investigate this particular scalability issue on the Tibidabo cluster by using a number of HPC codes from across a range of domains. The goal is to empirically evaluate which types of applications are suitable for that particular platform and what further optimisations from the hardware layer can be made to achieve a higher degree of performance at a lower energy footprint.

Finally, the Resilience, Fault Tolerance and Validation challenge is based on the in-fallible truth that hardware occasionally fails. Whether this is due to entropy or fault in manufacturing is irrelevant due to the fact that current programming models such as MPI used in parallelising HPC codes do not have any mechanism in place to cope with such event [40]. Furthermore, as one increases the number of components in a system, the Mean Time Before Failure (MTBF) increases dramatically and considering the fact that HPC systems already contain hundreds of thousands of cores, it is predicted that future Exascale supercomputers will have an MTBF of approximately 5 minutes [4]. This is unacceptable as it would make the entire system useless for any large problem sizes and simulations.

Even if fault tolerance mechanisms are put into place such as replication or by using naturally fault-tolerant algorithms [40], there is also the problem of validating the output of such a large system. After all, any insignificant manufacturing or design fault within the processing cores, whilst hidden at Petascale, could well be observed when moving to Exascale. If this becomes reality, all one has achieved is build a large supercomputer that can calculate wrong results three orders of magnitude faster than its predecessors.

One could however argue that by using embedded processors which are less complex in nature than their x86 counterparts mitigates this risk significantly. There is also the consideration that this design simplicity might allow for any potential errors to be spotted and rectified in time. However, no one has yet built a cluster using a million of these compute cores making any assumption on this matter a simple argument or opinion and it is without a doubt a consideration one needs to take into account.

In conclusion, even though there may be a number of intricate issues paved on the path to Exascale computing, reaching it is primordial for the continuous advancements in areas such as earth sciences, fundamental physics and astrophysics, nuclear energy research, life sciences and national security, all of which have a significant impact on human kind.

Perhaps the most important consideration that has to be taken into account when facing the Exascale challenge is that a solution must encompass and solve all of the above technological issues [33].

### 2.1.1 Mont Blanc

The Mont Blanc initiative is a project led by a consortium of establishments from both the academic environment and industry with the objective of developing an energy-efficient supercomputing capable of performing exascale computations [20]. Their goal
is to make use of commodity components from the mobile and embedded industry that already exhibit the necessary power draw characteristics needed for sustaining the operation of Exaflop supercomputers with the addition of a low price tag compared to bespoke architectures such as the one found in IBM’s Blue Gene series [14].

The project has set out three distinct milestones:

- Build an energy-efficient prototype by 2014.
- Provide further enhancements in order to achieve leading performance with a next-generation design for 2017.
- Optimise architecture to run a portfolio of Petascale applications.

Figure 2.1.1 presents the milestones pertaining to the architectural roadmap.

The first stage from figure 2.1.1 has already been reached with the inception of Tibidabo, a 256 node ARM-based cluster consisting of Q7 carrier modules developed by SECO [32] and an NVIDIA Tegra 2 SoC which is also used for the purpose of this project.

The aim for the second stage is to increase the performance density of the cluster with the addition of GPU acceleration in the form of a discrete NVIDIA GPU. This new iteration will be codenamed "Pedraforca" with predictions of a 35% efficiency improvement over conventional x86 + GPU solutions [14]. The overall computational performance is expected to be in the region of 120 Terraflops for 80 Kilowatt of power drawn.
The long term plan however in regards to the final architecture revolves around an integrated SoC containing both the necessary ARM-based processors with and an on-chip general purpose GPU sharing the same address space. This would all together remove the latency and power consumption inflicted by the bus communications between the conventional CPU + discrete GPU model and remove the power waisted on accesses to GDDR5 memory.

Perhaps the most interesting observation that can be derived from this initiative was made by the project’s technical coordinator, Dr. Alex Ramirez, during the European Exascale Way session at the International Supercomputing Conference’12. He wondered whether the “Mobile-killers” are coming referring specifically to the ARM-based processors used in smartphones today which currently lag behind in performance to x86 processors at a ratio of 1:8 but are clear winners when it comes down to cost being around two orders of magnitude less expensive.

Furthermore, due to the future advancements in the ARM architecture which is backed by a very profitable smartphone industry, tomorrow’s performance ratio could potentially be as little as 1:2 with the cost gap perhaps increasing to 1:1000. This in turn would draw a parallel to the specific period of time in the HPC ecosystem where vector computers were replaced by commodity microprocessors and exhibiting identical characteristics, less performance but lower cost and power consumption [14].

### 2.2 Energy Efficient Compute Architectures

#### 2.2.1 ARMed for HPC?

The prospective impact of ARM in the HPC market has been a recent talking point within the entire industry. This culminated during the International Supercomputing Conference’12 at the popular Analyst Crossfire session [18] where both HPC vendors and user representatives were asked upon their view on various current topics and trends pertaining to the supercomputing field.

One question that sparked the highest division among the four panel members regarded the number of ARM-based HPC systems breaking into the Top 500 list within the next three years and whether this number would be higher than twelve. Whilst two participants thought that the latter would be true, the other half disagreed with IBM’s VP of Exascale Computing, David Turek, going as far as predicting that no ARM-based clusters would be found in the Top 500 over the next three years.

One can of course suspect that his opinion holds a certain amount of bias bearing in mind that the company he is representing is behind the Blue Gene series which currently trumps the energy efficiency tables within the Top 500 Green list. Furthermore, there are certainly a large number of vendors who indeed fully back ARM’s involvement in the future of the supercomputing landscape such as NVIDIA with the inception of “Project
Denver" [19] and release of the CUDA on ARM development kit [17] culminating with their involvement in the Mont-Blanc initiative [20].

The reasons behind this backing are two fold. First of all, it is the advantage of the architecture over its x86 counterpart in terms of power efficiency. Sumit Gupta, NVIDIA’s Director of HPC products, postulated that the ARM processors are more energy efficient compared to Intel and AMD’s x86 variant due to the fact that the first was purposely built for the embedded world where the power wall was and always will be the primordial concern. On the other hand, x86 chips were always destined towards the PC market where there always was a greater need for branch prediction and speculative execution in order to cope with the unpredictable behaviour of the operating system and the user’s interaction with it such as mouse clicks [21][22]. For this reason, hardware engineers are forced to occupy the majority of the die area with such control mechanisms leaving a very small percentage for the actual compute [4].

For these reasons, replacing these speculative bundled cores with more simpler and compute oriented alternatives means that one can in fact still increase the core count on a die and achieve superior FLOPS per Watt performance [23]. This has certainly been a mantra adopted with great success by NVIDIA in the consumer graphics and gaming industry in their development of many core massively parallel Graphical Processing Units.

Whilst the difference in focus regarding power consumption has been true for a rather long time between the embedded and consumer market and hence the difference in design taken between the two platforms, a second reason for the growing interest in using ARM chips to power the next generation supercomputers are based upon the latest architectural developments.

For instance, the recent submission of patches to the Linux kernel for supporting the new AArch64 architecture based on the v8 platform is ARM’s answer for the progression to 64-bit which certainly provides great encouragement to its HPC supporters [24]. Although hardware is yet to be released using this new implementation, the new architecture means that an ARMv8 processor will be capable of rivalling with its x86 counterparts in terms of the size of addressable memory space.

The current 32-bit implementations of the chip restrict it to only 2GB of addressable main memory per core which is certainly an enormous drawback when moving into HPC as memory per core is suspected to grow into the future with current compute nodes hosting up to 64GB of main memory [18]. Furthermore, the transition to 64-bit brings forth other enhancements such as the increase in width for all general purpose registers to 64-bit and improvements to vector operations with the replacement of the previous 32 64-bit registers that could be aliased into 16 128-bit pseudo-registers to 32 128-bit registers which essentially doubles the capacity of the vector unit and will increase the performance of SIMD operations also known as NEON [25].

A negative point however is the fact that multi-threading support is still not present. This is put down perhaps to issues in validating such a large overhaul of the entire platform. What this means is that the ARMv8 chip will still not be able to fully compete with
other RISC-based processors such as the PowerPC A2 found in the Blue Gene/Q series which has recorded increased performance when running in multi-threaded mode will full occupancy [26].

There is however a consideration to take into account. The release of the ARMv8 64bit architecture is only scheduled for early 2014. Whilst this may seem like a long wait for the rapidly changing HPC ecosystem, further good news have arrived on the ARM’s Mali GPU front which comes packed into its SoC package for the mobile space. This was presented in the form of a press release by ARM’s GPU Computing Marketing Manager which announced the submission to Khronos [27] for OpenCL 1.1 Full Profile conformance [28].

In essence, what this means is that with Full Profile, all features of the OpenCL implementation that were available on desktops, laptops and servers are going to be supported by the on-chip GPU device such as:

- Native support for 64-bit maths including vector operations on any Mali-T600 series GPUs.
- Compliance to IEEE 754-2008 precision making double precision as accurate as on all other platforms that conform to the Full Profile standard.
- Addition of atomic functions that are implemented and accelerated in hardware which means synchronization in parallel computing can now be done more efficiently.
Figure 2.2.1 presents the newest addition to the Mali T600 series, the T678, that will benefit from full OpenCL 1.1 support. This particular implementation contains eight shader cores used for processing split into two blocks of four each with their own L2 cache.

One could easily envisage how floating point operations could be off-loaded to this unit within the SoC leaving the low-powered CPU’s such as the Cortex-A9 [29] to deal with communication tasks et al. This particular programming model could not be easily achieved before due to the fact that all on-chip GPU’s only supported OpenGL ES which is targeted specifically for graphics programming making general purpose compute very cumbersome [14].

Having all of these building blocks under the same memory address space means that there is no latency penalty of going over buses such as PCIe that is usually encountered in the CPU and discrete GPU model. Furthermore, due to OpenCL’s cross-platform standard, a new heterogeneous architecture could be put in place where highly computational kernels could still be sent to a more powerful discrete GPU whilst routine computation can be performed on the low-powered on-chip counterpart therefore minimizing data movement and power draw over an increase in performance and compute efficiency.

2.2.2 CUDA on ARM

CUDA on ARM is a newly released platform carrying the codename "CARMA" designed by NVIDIA as a prototype for energy efficient hybrid computing in the high performance computing field [17]. The architecture consists of:

- Q7 carrier board produced by SECO containing peripherals such as USB, 4xPCIe Gen1 and Gigabit Ethernet Interface [32].

- NVIDIA Tegra 3 SoC.

- NVIDIA Quadro 1000M GPU with 96 CUDA cores.

The purpose of the Carma platform is to build a community of developers and researchers around the ARM + CUDA ecosystem which is envisaged by NVIDIA to be a pre-incarnation of the future HPC system [41].

Internally, the low power host CPU found in the Tegra 3 SoC is composed of 4 ARM A9 cores that are capable to execute NEON instructions and VFPv3 extensions.

Furthermore, Tegra 3 boasts a number improvements over the 2nd generation SoC used in Mont Blanc’s Tibidabo prototype. Some of these are:

- 2x increase in number of cores and frequency (up to 1.3GHz from 1GHz).

- Lower power utilisation with vSMP.

- 2-6x Faster access to storage devices.
• 3x faster integrated GPU although still not support for general purpose program-
ming.

CUDA GPU Tegra ARM CPU

Figure 2.3: Carma Development Kit

The Quadro1000m discrete GPU that is provided with the first generation of the board contains 96 CUDA cores with 200 GFLOPS single precision peak performance. Upon questioning however, Don Becker of NVIDIA stated that as double precision performance goes, one should expect approximately 20-30 GFLOPS for this particular device as its primary development targeted portable devices such as netbooks and ultrabooks [41].

The software stack consists of a Ubuntu 11.04 distribution based on the ARM architecture and supporting the 3.1.10 Linux kernel version with enhancements to support Tegra SoC devices. Programming the discrete GPU will be possible using a ported version of the CUDA 4.2 run-time including all its supported libraries however one must cross-compile the code from an x86 host. Future enhancements will see native compile support and an update to the newest 5.0 CUDA run-time. Furthermore, the long term plan is to support the future release of the ARMv8 64bit architecture which has been described in more details in the above section [41].

As earlier mentioned, the project’s initial goal was to develop a small-sized cluster based on this new platform and pit it against Tibidabo in order to empirically evaluate the increase in computational efficiency brought forth with the addition of a discrete GPU. Even though this was not possible due to delays in the manufacturing and shipment of the board, the prospects look encouraging.

Perhaps the most important thing to keep in mind is that the architecture supports the inclusion of any GPU in MxM form which could also include NVIDIA’s newest Kepler GPU [42]. This is a very important factor to take into account since replacing the
Quadro1000m GPU with a more advanced Kepler-based device could deliver orders of magnitude increase in computational performance. The main reasons for that being the new improvements that were incorporated to this new platform such as:

- New power-aware SMX Architecture.
- Increased register file (255 registers per thread).
- New atomic instructions.
- Dynamic Parallelism.
- Hyper-Q.

With Dynamic Parallelism, as seen in Figure 2.2.2, the GPU can essentially create work for itself therefore being able to keep very expensive computational kernels on the device without the need of repeated communication with the host CPU over the expensive PCIe bus. This also means that the CPU is freed to perform other tasks such as MPI communications therefore achieving a superior form of overlap between computation and communication [43].

![Figure 2.4: NVIDIA Kepler - Dynamic Parallelism](image)

The new Hyper-Q mechanism illustrated in Figure 2.2.2 allows for all cores contained in the host CPU to schedule work on the GPU device simultaneously. This is a major
improvement over the previous Fermi architecture where only one core could perform computations on the device at a single time and results in the possible full exploitation of the Kepler’s computational performance [43].

Figure 2.5: NVIDIA Kepler - Hyper-Q

All of these features are incorporated on a single chip built out of 7.1B transistors with a peak 64-bit floating point precision of over 1 TFlop [42].

One can therefore postulate that the Carma platform has all the necessary ingredients of an energy efficient compute architecture to be used in future highly energy efficient HPC systems.

2.2.3 Project Denver

In order to facilitate ARM’s coverage of a broader spectrum of the computing ecosystem, NVIDIA announced their initiative to enter the CPU market by developing a high performance ARM-based processor under the codename "Project Denver" [19].

The main aim of this endeavour is to create a synergy between a low-powered ARM CPU and a powerful energy efficient NVIDIA GPU on a single die and governed by the same instruction set. Although there are no further details on the architecture’s overview [30], NVIDIA’s Bill Dally argues that the new chip will overcome all the power and performance issues of x86 CPU’s by bringing forth heterogeneity in the design [19].

Up until now, a heterogeneous system usually consisted of a powerful multi-core x86 CPU and an accelerator located at the other hand of a PCIe bus [31]. Programming the two required different programming models and languages due to the discrepancies in the instruction set used. This would not be the case within the future NVIDIA CPU
as all compute building blocks (CPU and GPU) would use the same ISA and global memory space therefore delivering a new breed of hybrid systems [25].

In terms of power efficiency however, it is envisaged that dynamic thread scheduling could be performed in order to migrate tasks from the low-powered ARM cores to the GPU counterparts depending on necessity and advantage in performance over watt that would be obtained. This would mean that voltage and frequency scaling would be a usual phenomena within the chip as certain parts power down and back up depending on whether any tasks are assigned to them. Although the release of this particular processor is scheduled for 2013, it will be exciting to see whether the prospective advantages that this platform would offer will indeed become a tangible reality.

2.2.4 Project Echelon

In an effort to facilitate the development of novel energy efficient compute architectures, the Defense Advanced Research Projects Agency (DARPA) has created the "Ubiquitous High Performance Computing" [44]. The first phase of this initiative is to allocate funding to a number of collaborations formed by major HPC vendors and research institutions who’s goal is to conduct preliminary research on how to overcome the technological barriers that lay in front of the transition to Exascale computing [45].

One of the beneficiaries of this endeavour is "Echelon", a project led by NVIDIA and consisting of a partnership with a number of other entities such as: University of Utah, University of Pennsylvania, CRAY, Oak Ridge National Laboratories, Micron, Lockheed Martin, University of Texas, University of Tennessee, Georgia Institute of Technology, University of California and Stanford University [48].

The fruits of this collaboration have been witnessed at the Supercomputing 2010 conference where NVIDIA's chief scientist, William Dally, offered a sneak peak to a mock-up architectural design (Figure 2.2.4) of a 10TFlop processor which according to him would serve as the building block for Exascale-class supercomputers [45][46][47].

A more in depth view of the architecture proposed in Echelon was given by Dr. Timothy Lanfear of NVIDIA in a guest lecture at the Edinburgh Parallel Computing Centre [48] where he described the principles on which this platform was designed. He argued that the main source of power draw in current systems is data movement as evidenced in Figure 2.2.4 where fetching an operand incurs a higher power penalty than computing on it due to the traversal of the entire width of the 28mm die in order to reach main memory which subsequently results in almost a three orders of magnitude increase in power draw.
The Echelon processor aims to solve this issue by keeping data local to the 128 Streaming Multiprocessor (SM) compute cores (256 under the revised design [49]) for as long as possible through a large switched shared L1 cache and 1.024 banks of 256KB L2 static RAM cache (256GB in total) therefore delivering a more optimized memory hierarchy [45].

One of the interesting features of this chip is the concept of "malleable memory hier-
archy" which is achieved through the custom Network On-Chip (NoC) controller that can restructure the memory hierarchy depending on the type of data access pattern performed by the running application [49]. For example, when computing a double precision general matrix multiply (DGEMM), the NoC can be reconfigured to allow for a deep cache hierarchy in order to maintain the necessary data on-chip whilst a table access pattern would be provided with a shallow cache structure to further minimize unnecessary data movement from the more slower higher level caches [48].

The importance of having a reconfigurable memory hierarchy is further evidenced in Figure 2.2.4. Although there are certain numerical algorithms such as FFT’s and Ray-tracing that do not exhibit a large energy discrepancy based on the underlying storage hierarchy, a considerable difference can be seen when performing matrix multiplications or volume renderings [50].

![Flat Vs Hierarchical Energy (Normalized)](image)

**Figure 2.8: Flat vs. Hierarchical Memory**

There is no doubt that the magic formula envisaged by Echelon (Figure 2.2.4) for the future of energy efficient compute architectures is based on a more evolved expression of heterogeneity through the coupling of two architecturally different components such as the Streaming Multiprocessors (SMs) and Local Compute (LC) cores. It is predicted that the latter shall be based on the ARM platform due to its low power consumption
nature and dispatched with running the node’s operating system and software stack [48]. Furthermore, gluing both of these compute elements on the same die and under a global memory address space removes the unnecessary power consumption and latency penalty of current heterogeneous platforms since the communication between the two does not have to travel off-chip any more via slow buses such as PCIe.

The final Exascale system based on this prospective architecture would see a multi-chip node module containing eight Echelon processors on a single substrate having access to 3D stacked DRAM memory and a 150GB/sec bandwidth via the Dragonfly interconnect for inter-node communication. A cabinet would further consist of 16 modules with a predicted performance of 2.56 PFlops for 38 KW of power draw [45]. Finally, approximately 400 of these cabinets would suffice to reach an Exaflop performance at approximately 15 MW [50].

In conclusion, although the Echelon architecture brings forth highly intelligent concepts for improving the computational efficiency needed for Exascale, it is important to note that the Echelon processor is still within the design phase. More importantly, the intricacies of manufacturing such a chip are not to be underestimated not mentioning the cost. The latter is a very important factor to take into account. NVIDIA recently announced that the research and development price tag for delivering its new Kepler architecture was over 1B $ [43]. In Kepler’s case, the cost will most likely be covered by major sales in the profitable gaming and consumer markets. This would not be the case for Echelon since the targeted HPC market is substantially smaller with tighter profit margins compared to the other two. Unless the project manages to marry this concept with its "Project Denver" architecture and use it as a foundation for the "future universal chip", it is highly unlikely that one would ever witness the Echelon processor "in the wild" despite its tremendous features.

2.2.5 BlueGene/Q

The IBM Blue Gene/Q’s Power BQC chip is currently the most energy efficient compute architecture to date dominating the Top 500 Green list [13]. It follows the same approach as its previous generations such as the Blue Gene/L and Blue Gene/P where a great care and consideration is given from the very beginning to price performance, energy efficiency and reliability [26].

One of the reasons for the BQC’s success on following and improving this philosophy is through the utilisation of power efficient processors which allow for a more dense packaging resulting in a power, cost and floor-space efficient architecture [51]. Perhaps one of the most surprising thing that one can observe in the BQC chip is that there are no revolutionary components or manufacturing technologies that give it its superior computational efficiency compared to the rest of today’s Top500 supercomputers. In fact, the chip is based on the PowerPC architecture which has been powering computing devices for over two decades. The difference however is made on deploying an embedded System on-Chip (SoC) design which leads to a very significant decrease in
complexity [51]. Adhering to simple design principle allows the Blue Gene/Q to scale to more than a million processors whilst still maintaining high reliability and compute efficiency.

Inside, the BQC chip (see Figure 2.2.5) contains 18 processing units based on the PowerPC A2 core with additional implementations such as quad floating point, wake-up and L1 pre-fetching units [26]. The user is only exposed to 16 of these cores leaving one of the other two in charge of running the operating system with one core redundant to increase manufacturing yields [51]. Each PowerPC A2 core is clocked at 1.6 GHz and supports 4-way Symmetric Multi Threading (SMT) achieving a peak computational performance of 204.8 GigaFLOPS [51]. Access to L2 cache is via the "xbar switch" as seen in the centre of the die with the former being 32 MBytes in size and supporting speculative execution as well as transactional memory [26].

![Figure 2.9: Blue Gene/Q Power BQC architecture](image)
The elements that make this particular chip stand out from a regular PowerPC A2 processor are the aforementioned additions that were specifically implemented for achieving increased performance and efficiency through the process of co-design.

The quad floating-point processing unit attached to each core is based on the Quad Processing eXtension (QPX) added to the ISA of the Power family and defines a new set of instructions for floating-point operations therefore replacing the traditional FPU from the generic chip version [26]. This new implementation can perform up to eight double precision floating-point operations per cycle through the 4 available execution slots each capable of storing 256-bit vectors. The result is a peak computational performance for each core of approximately 12 Gflops [26].

The L1 pre-fetching unit is in charge of hiding the latency of accessing the L2 cache and can execute two different types of data pre-fetching such as stream pre-fetch and list pre-fetch by storing the data into a coherent buffer. The stream pre-fetch executes the standard approach of similar engines by identifying the next block of data needed when traversing contiguous memory and loads it into the buffer before it is requested. The list pre-fetch mode is more advanced as it can anticipate the sequence of loads even from an arbitrary set of addresses when the particular access pattern is repeated [26].

Finally, the wake-up unit is a mechanism incorporated in order to reduce the time it takes for the hardware to dispatch a task to an executing thread [51].

It is important to note that all of the above additions have been incorporated and designed with scalability in mind with a special emphasis on Quantum Chromodynamics applications. The new QPX mechanisms allows for paired complex operations to be performed since it can store quad doubles in one execution slot whilst the L1 pre-fetching units fit well with the access pattern one encounters in these types of applications [52]. The wake-up unit however has more of a generic target since minimizing thread context switching and has the same positive effect on all applications that make use of SMT.

In conclusion, the success of the Blue Gene/Q seems to revolve around its design principle. Using a relatively simple low-powered SoC as building block and providing further enhancements such as the ones described above resulted in the inception of the world’s fastest and most energy efficient supercomputers.

This approach should be taken as an example for the future trajectory of ARM-based compute architectures which could guarantee its dominance of the HPC ecosystem bearing in mind the distinct advantage that it has over the PowerPC architecture in terms of popularity and openness. Furthermore, using the ARM architecture would allow each vendor to tailor their design to particular applications or use cases through co-design.
2.2.6 Intel Xeon Phi

The Intel Xeon Phi coprocessor which was officially announced at the International Supercomputing Conference’12 [53] is Intel’s first device targeting the HPC segment and featuring the many integrated cores architecture (MIC) [54]. It signifies Intel’s change of perspective as it makes the transition from the previous homogeneous x86 multi-core chip architecture to heterogeneity where a powerful Xeon processor uses the many core Xeon Phi coprocessor as accelerator for highly parallel work loads [56].

The Xeon Phi chip (see Figure 2.2.6) is designed to integrate over 50 simple x86 processing cores under the new Knights Corner platform, each with a 4-way SMT and 512 bit-wide SIMD registers [55]. The previous incarnation of this device, namely the Knights Ferry, managed to obtain 1 TFlop performance on the SGEEM benchmark containing only 32 cores on a die [57].

![Figure 2.10: Intel Xeon Phi Die](image)

Although the product was not officially released yet with prototypes only provided to a number of partner institutions, Intel announced that it will include Xeon Phi coprocessors in Texas Advanced Computing Centre (TACC) 10 Pflop machine codenamed "Stampede" [54].

According to Intel, the advantage of this architecture is that it allows the usage of a variety of programming models when targeting the device. The Xeon Phi can function as both a coprocessor via the PCIe bus forming a virtual network with a Xeon-based multi-core CPU or as a stand-alone device due to the underlying x86 architecture which gives it greater flexibility. Furthermore, it argues that any existing codes that already run on x86-based homogeneous platforms can be ported to the MIC architecture via a
simple recompilation process using the provided software tools.

Figure 2.2.6 illustrates the Xeon Phi processing card to be deployed in future supercomputers such as TACC’s "Stampede".

![Figure 2.11: Intel Xeon Phi Card](image)

Although the prospective ease of programmability that this architecture may offer sounds very appealing to the HPC community which has had to battle with a fluctuation of highly intricate programming models over the years, there is no real data available regarding the power efficiency of the Xeon Phi nor any substantial performance results other than the "Discovery" HPC cluster which managed to break into the Top 500 list on position 150 [59].

This is also partly due to the non-disclosure commitments that Intel’s partners signed which prohibit them from revealing any of the performance data gathered through their initial research on the device. Furthermore, NVIDIA’s CTO Steve Scott argues that the MIC architecture will not be able to provide the proclaimed "free lunch" to the HPC ecosystem in terms of obtaining dramatic performance increase by simply recompiling the code for this architecture [58]. The main projected argument is that one cannot optimise a processing core for both energy-efficiency and single-thread performance as they tend to be mutually exclusive. Therefore, offloading the entire application to the MIC will bring forth even more issues such as network congestion when using MPI as the network traffic would have to travel through PCIe to the network adapter or poor memory utilisation in the case of OpenMP due to Ahmdahl’s Law and insufficient expressed parallelism [58].
It seems therefore that in order to assert the full characteristics and performance of the Xeon Phi, one must have to wait until "Stampede’s" performance metrics for both power consumption and computational performance are released so as to make a fair appraisal of this architecture and its subsequent future in HPC.

2.2.7 Summary

This chapter provided an overview on the Exascale challenge together with the potential influence that ARM-based processors might have on the future of the supercomputing landscape by presenting its architectural roadmap and comparison with current or future energy efficient compute architectures.

The next chapter will present an overview of the Tibidabo cluster used in this project as the primary benchmarking target and the focus of the main investigation.
Chapter 3

Tibidabo

This chapter presents a description of the Tibidabo ARM multi-core HPC cluster developed by the Barcelona Supercomputing Centre under the Mont-Blanc project and the target of this project’s investigation into power efficient compute architectures.

3.1 Architecture Overview

The latest incarnation of the Tibidabo prototype consists of 256 nodes split evenly across two racks out of which a total of 248 are compute nodes and 8 login nodes. Each node is based on SECO’s Q7 module and carrier board and hosts the following components:

- **Q7 module**
  - 1 NVIDIA Tegra 2 SoC (Harmony)
  - 1 GB DDR2 DRAM
  - 100 MBit Ethernet
  - PCIe (1 GbE and GPU MxM interface)

- **Q7 carrier board**
  - 2 HDMI ports (1x GPU and 1x CPU)
  - uSD slot
  - 2x USB ports
  - 8" x 5.6" size
The NVIDIA Tegra 2 "Harmony" SoC (see Figure 3.1) contains:

- 2x Cortex-A9 processing cores clocked at 1 GHz
- Low power NVIDIA GeForce ULP GPU with 4 vertex and 4 shader cores clocked at 333 MHz
- Video, Audio and Image processing accelerators
- ARMv7 architecture

The chip is capable of approximately 2 GFLOPS single precision at 0.5 Watt power draw and contains a vector floating point unit (VFP) although there is no support for ARM’s NEON instruction set therefore limiting it’s SIMD processing capabilities. Furthermore, the on-chip GPU only supports OpenGL ES 2.0 as programming interface which leads to non-trivial utilisation for general purpose compute [61].

![Figure 3.1: NVIDIA Tegra 2 SoC](image)

A Tibidabo blade (multi-board container) (see Figure 3.1) is based on standard 19" rack dimensions and hosts 8x daisy-chained Q7 carrier boards each containing 8 Tegra 2 SoC and 8 GB of DRAM with 1 power supply unit and averages approximately 16 GFLOPS single precision performance for 40 Watts power draw [62].

The prototype rack (see Figure 3.1) is set up to hold 8 of these blades with one Ethernet switch being provided per 4 compute nodes (i.e. Q7 carrier boards). External cooling of the system is not available with the brunt of the work being performed by the heat sink attached to the Q7 module. Power consumption measurements can be performed per unit, blade or entire rack although the latter is only true at the time of writing due to configuration issues.
Finally, the cluster utilises both Network Interface Cards with the 100MbE provided by the carrier board being used for the service network in supporting the NFSv3 file system.
whilst the 1Gbe interface supported by the PCIe interface is used for data communications such as MPI. The overall infrastructure uses SF200-48 CISCO Gigabit Switches with the nodes being connected in a Tree-like topology [60].

3.2 Software Overview

The software stack (Figure 3.2) running on top of the hardware layer is fully open source and consists of [62]:

- Linux OS (2.6.32)
- Runtime Schedulers (Slurm, Sun Grid Engine)
- GNU Compiler Collection (gcc 2.4.6 and gfortran)
- Scientific Libraries (FFTW, ATLAS)
- Runtime libraries (OpenMPI, MPICH2, OmpSs)
- Profilers and Debuggers (Paraver, Scalasca, Allinea DDT 3.1)

Figure 3.4: Tibidabo Software Environment

The default scheduler for Tibidabo is Slurm although three wrapper scripts are provided for interfacing with it such as:
mnq - displaying the queue status
mnssubmit - submitting a job to the queue
mncancel - cancelling a job in the queue

Access to the full Slurm utilities can be done via updating the PATH environment variable. These are located in the /opt/perf/bin/ directory however the functionality of the provided wrapper scripts was sufficient for the purpose of this project.

3.3 Power Efficiency

Preliminary investigation into the computational efficiency of the Tibidabo cluster unveiled interesting characteristics [14]. Although the Tegra 2 SoC contains low-powered ARM Cortex-A9 cores, their modest computational performance means that the main power draw on a node is actually attributed to the other components hosted by the Q7 carrier board as seen in Figure 3.3. As a result, there is an evident detrimental effect on the computational efficiency of this platform which would perhaps require either the addition of a higher density multi-core chip or a power efficient accelerator such as a discrete GPU which should dramatically increase the node’s performance per Watt [14].

Furthermore, it will be interesting to empirically evaluate the possible oscillations in Tibidabo’s power draw when running different classes of applications. For instance, one would assume that a QCD application would have a lower computational efficiency than a Monte Carlo simulation bearing in mind that the former would utilise the node’s GbE adapter more frequently and therefore increasing the power consumption on a node.

![Figure 3.5: Tibidabo Node Power Consumption](image-url)
There is also the consideration that this architecture was created as an initial prototype and spring board for ARM’s entry into the HPC ecosystem. It is highly likely that a more customized carrier board that would reduce the number of unnecessary peripherals such as HDMI interfaces et al. would contribute to the reduction of the currently waisted power.

3.4 Issues

The team in charge of assembling and fine-tuning the Tibidabo prototype have shared some of the lessons learned through their experience [63]. The described problems varied from optimising the underlying operating system to choosing whether to use software-based floating point vector units vs. their hardware counterpart.

Working on a low-powered architecture meant that compiling and optimising scientific libraries such as ATLAS on the ARM Cortex-A9 chip took over 1 month. A large amount of time was also needed for the manual labour of assembling the cluster with two persons spending one month simply pressing all the needed screws [63]. Furthermore, one of the more important realisations was that even low-power devices need some form of cooling whether this is passive or active and that the main culprit for the majority of the system’s power draw in an embedded architecture is in fact memory and not the CPU.

3.5 Summary

This chapter outlined the main features and characteristics of the Tibidabo prototype and provided information on its architecture, preliminary power metrics and software environment.

The next chapter contains a description of the empirical methods that were used to further evaluate its performance through the execution of synthetic benchmarks and real HPC applications.
Chapter 4

Benchmarks

This chapter presents a description and critical review of the evaluation methods used in benchmarking the Tibidabo cluster.

4.1 Evaluation methods

Empirically assessing the computational performance of HPC systems and quantifying this into a metric is common practice in the supercomputing field in order to allow for a distinction and classification between each system. This is usually achieved by running the now popular High Performance Linpack benchmark which solves a system of dense linear algebra equations and can provide an overview on the peak computational performance of the system. The Top500 list comprising of the world’s most powerful supercomputers together with its Top500 Green alternative focused on energy-efficiency use HPL as the de-facto benchmark for its published hierarchy [64].

There are however certain aspects one has to take into account when assessing a supercomputer’s capabilities solely on HPL or indeed any synthetic benchmark. First of all, they do not pertinently reflect the wide range of differences that exist in common real HPC applications in use today. For example, applications can inherit different characteristics that might be able to fully exploit the underlying architecture or on the contrary, lead to very poor performance results.

There are a number of system characteristics that can affect this behaviour such as memory hierarchy as originally described in the "Project Echelon" initiative where a wide class of numerical methods perform better with different cache hierarchies both in terms of computational performance and energy efficiency [48]. Another such example is the different algorithmic nature implemented in HPC applications. For example, a Monte Carlo simulation is usually regarded as compute bound since the main bottleneck lies with the CPU’s inability of generating random numbers at a fast enough interval therefore stalling overall performance and progress. A parallel multi-dimensional FFT algorithm however tends to exhibit a very high amount of communication over computation
due to the constant need of re-arranging the data in each dimension which can affect scalability as the impact that these collective operations have on performance increases together with the core count.

Finally, a high number of applications incorporate all of the above with examples such as Quantum Chromodynamics (QCD) where both bottlenecks in computation and communication can be present through the utilisation of hybrid Monte Carlo methods and Molecular Dynamics.

Perhaps the most conclusive example that can be illustrated to support all of the above can be found in Dr. Peter Boyle’s recent presentation at the International Supercomputing Conference’12 on the topic of QCD special purpose compute machines such as the Blue Gene/Q which includes components that were specifically added to increase the performance of these types of applications [52]. He presented the results from running an optimised QCD application that made full use of Blue Gene/Q’s interconnect, QPX and L1 pre-fetch units on half of the Sequoia supercomputer’s cores (500,000) which resulted in 3 PFlop sustained performance [52]. Interestingly though, the HPL peak computational performance of the full system is approximately 16 PFlops making it the world’s most powerful HPC system. Bearing in mind that HPL is known to scale well as one increases the problem size with the number of cores and memory, one could postulate that the peak computational performance of half the system would be in the region of 8 PFlops. In conclusion, there is a very large discrepancy between the performance metrics obtained through running HPL compared to this particular QCD example whilst also bearing in mind the co-design effort [51] that has made the latter record such impressive results.

In conclusion, in order to portray a more accurate evaluation of the Tibidabo system, a range of synthetic benchmarks and real HPC applications will be used. The first is useful for obtaining performance metrics on certain system specifics characteristics such as memory and interconnect performance whilst the latter will focus on asserting the scalability of different classes of applications on the ARM-based compute nodes. An emphasis will also be placed on energy efficiency with power consumption metrics also being provided.

### 4.2 Synthetic Benchmarks

#### 4.2.1 HPC Challenge Benchmark

The High Performance Computing Challenge Benchmark is based on 7 distinct tests which aim to assess the overall capabilities of the system [66] such as:

- **HPL**: solves linear system of equations measuring peak floating point computation performance.
- **DGEMM**: similar to HPL but focuses on measuring double precision floating point performance.
point performance by solving general matrix on matrix multiplications.

- **PTRANS**: asserts the overall capacity of the interconnect by executing matrix transposes across all processors.

- **RandomAccess**: as the name implies, records the performance of random integer accesses to main memory.

- **FFT**: performance of double precision complex one-dimensional Discrete Fourier Transforms computations [66].

- **Ping-Pong**: message around a ring benchmark although also including randomised version for fully testing the latency of the network.

The main motivation behind the elaboration of a suite of benchmarks is that of quantifying the overall performance of a system more accurately. The distinct properties of each benchmark within HPCC gives their combine effort a more realistic scenario of evaluation in relation to real applications codes. For instance, the DGEMM and HPL exhibit high temporal and spatial locality therefore being very suitable for testing the computational peak performance of the system’s CPU’s. The FFT benchmark on the other hand whilst showcasing good temporal locality, it performs poor in terms of data reuse therefore being suitable to test the characteristics of the system’s cache hierarchy. Another example is the RandomAccess test which has poor spatial and temporal locality and sets out to probe the capabilities of the system for accessing random data, a pattern which is present in a number of HPC applications [64].

In order to put the results of HPCC into context, a comparison with other similar systems will be made that have executed the benchmark and officially submitted their results to the HPC Challenge competition website [66].

### 4.3 Application Benchmarks

The HPC applications used for the benchmarking task have either been developed, ported or fixed in order to run on the Tibidabo cluster as part of this project. A brief description of each one of them shall be presented below in regards to their characteristics and purpose.

#### 4.3.1 Jacobi Iterative Solver

A parallel image processing application has been re-developed from scratch for the purpose of this project which applies the inverse of a simple edge detection algorithm in order to fully reconstruct the initial image. There are two available implementations which can decompose the image data across processes in either 1D or 2D.

In 1D mode, the program reads the image and scatters a slice of it to each processor in the pool using a contiguous derived data type and a subsequent vector data type for
avoiding the halo regions in the buffer. In the 2D example, the application decomposes the image across all processors in a specially created Cartesian topology by scattering across rows first followed by columns.

Halo swapping is performed in a non-blocking manner in order to allow for the overlap of computations and communications in each iteration with the convergence value being calculated at specific intervals. Finally, the data is being gathered back together on the master rank following the inverse steps depending on the chosen decomposition.

This application is particularly suitable for analysing both the computational and communication aptitudes of a system due to the existence of overlap between calculating the individual pixel values and receiving the necessary halos. An unbalanced system will not achieve great scalability nor speedup by running on a high core count due to either the CPU being too fast for the interconnect or potentially vice versa.

### 4.3.2 Black Scholes Option Pricing

The Black-Scholes Option Pricing code has been ported from C++ to C and parallelised using MPI. The application is used to simulate the future price of an option given the initial price of the stock and a future point in time using the Black-Scholes equation and Monte Carlo simulations. The specific purpose of the latter is to model the random behaviour of stock which is known to follow the characteristics of the Brownian motion.

The program reads an initial input file containing 1024 options and distributes it evenly across the processing pool by using a collective scatter and an MPI derived data type for contiguous memory mapping. After all necessary computations are performed, the master gathers all the options and prints the future price of each option depending on the provided parameters.

This particular application is compute bound since there is no communication needed after the initial distribution of the work load with performance varying depending on the CPU’s capability of generating the random numbers needed by the Monte Carlo simulation at a quick enough rate.

### 4.3.3 Particle Simulation

The particle simulation application is a parallel MPI-based implementation which simulates the motion of plasma charged particles that interact with one another electrostatically with the source code being provided through the Indiana University course pages [67].

The application exhibits some of the characteristics found in Molecular Dynamics codes where after each time step in the simulation, the processors swap the particles with each other in order to calculate the necessary forces and subsequent position on the three dimensions.
A small bug fix has been provided to the code for this particular project due to its old nature and not conforming to the MPI-2 standard.

Executing this simulation shall provide insight into the overall balance of a system in terms of double precision compute capabilities and interconnect performance, both of which will have an impact on scalability and speedup.

### 4.4 Summary

This chapter has presented the evaluation methods to be used for the empirical performance assessment of the Tibidabo ARM-based cluster. A discussion on both synthetic and real application benchmarking has been given together with the underlying motive behind using both for the evaluation process.

The next chapter shall illustrate the results and analysis that stand as proof for the execution of the above.
Chapter 5

Results and Analysis

This section shall present the results of the aforementioned benchmarks that were run on the Tibidabo system and their subsequent analysis under the project scope.

5.1 Compiling Optimisations

In order to allow for a fair portrayal of the performance of the Tibidabo cluster and the Tegra 2 SoC, an initial investigation has been carried out with the purpose of finding the optimal gcc compilation flags that should be used when building all benchmarks irrespective of their nature. The results of this initial test can be seen in Table 5.1.

<table>
<thead>
<tr>
<th>Compiler Flags</th>
<th>Execution Time (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>-mcpu=cortex-a9 -mfpu=vfpv3-d16 -O3</td>
<td>173.3</td>
</tr>
<tr>
<td>-mfpu=vfpv3-d16 -O3</td>
<td>174.2</td>
</tr>
<tr>
<td>-mcpu=cortex-a9 -O3</td>
<td>174.5</td>
</tr>
<tr>
<td>-mfpu=vfp -O3</td>
<td>175.7</td>
</tr>
<tr>
<td>-O2</td>
<td>179.7</td>
</tr>
<tr>
<td>-O0</td>
<td>208.2</td>
</tr>
</tbody>
</table>

Table 5.1: GCC ARM Optimisation Flags for serial MD code

The above timings have been obtained from the serial execution of a Molecular Dynamics code which calculates the inverse-square gravitational force between particles using Newton’s Law of Motion. This example was chosen due to the fact that it exhibits a wide range of operations such as double precision floating operations and local and sparse memory accesses.

As one can observe from the above results, the best performance was obtained when specifying both the -mfpu and -mcpu flags which allow the GCC ARM compiler to
further optimise the code using Cortex-A9 specific instructions and optimised vector floating point unit instructions for the FPU.

5.2 HPC Challenge Benchmark

5.2.1 HPL

The performance of the HPL benchmark can be seen in Figure 5.2.1. As one can observe, increasing the number of nodes and cores together with the problem size yields an increase in performance.

![High Performance LINPACK](image)

**Figure 5.1: High Performance LINPACK Performance**

Due to various maintenance work to the Tibidabo system, the maximum core count available for running HPL was 212 which coupled with a problem size of 99687 obtained a 103.3 GFlops peak performance. This can be witnessed in Table 5.2.1 including the relevant Flops per Watt metrics that were calculated after measuring the average power utilisation during the benchmark run.
Table 5.2: HPL Performance on maximum core count

<table>
<thead>
<tr>
<th>No. of Cores</th>
<th>GFlops/s</th>
<th>Power (Watts)</th>
<th>MFlops/Watt</th>
</tr>
</thead>
<tbody>
<tr>
<td>212</td>
<td>103.3</td>
<td>502.99</td>
<td>205.371</td>
</tr>
</tbody>
</table>

Figure 5.2: HPL Power Consumption

Based on these results, Tibidabo would officially occupy position 226 (see Figure 5.2.1 in the Top 500 Green list and therefore achieving it’s 200MFlops per Watt target which was set out for the prototype within the Mont-Blanc project.

Figure 5.3: Tibidabo position in Green Top 500

More details on the HPL.dat parameters used for each run can be seen in the Appendix section of this report.
5.2.2 DGEMM

<table>
<thead>
<tr>
<th></th>
<th>Star DGEMM</th>
<th>Single DGEMM</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.592708</td>
<td>0.623177</td>
</tr>
</tbody>
</table>

Table 5.3: DGEMM Benchmark in GFlops/s

5.2.3 STREAM

<table>
<thead>
<tr>
<th></th>
<th>Star STREAM Copy</th>
<th>Star STREAM Scale</th>
<th>Star STREAM Add</th>
<th>Star STREAM Triad</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.775263</td>
<td>0.345928</td>
<td>0.313078</td>
<td>0.204907</td>
</tr>
</tbody>
</table>

Table 5.4: Star Stream Benchmark with vector size 205761

<table>
<thead>
<tr>
<th></th>
<th>Sin. STREAM Copy</th>
<th>Sin. STREAM Scale</th>
<th>Sin. STREAM Add</th>
<th>Sin. STREAM Triad</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.933378</td>
<td>0.346179</td>
<td>0.315827</td>
<td>0.206138</td>
</tr>
</tbody>
</table>

Table 5.5: Single Stream Benchmark with vector size 205761

5.2.4 PTRANS

<table>
<thead>
<tr>
<th></th>
<th>GB/s</th>
<th>Time (seconds)</th>
<th>N</th>
<th>NB</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.346718</td>
<td>0.576838</td>
<td>5000</td>
<td>116</td>
</tr>
</tbody>
</table>

Table 5.6: PTRANS Benchmark Results

5.2.5 RandomAccess

<table>
<thead>
<tr>
<th></th>
<th>Star RandomAccess</th>
<th>Single RandomAccess</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.00632323</td>
<td>0.00764762</td>
</tr>
</tbody>
</table>

Table 5.7: GUPs Star and Single Random Access N=524288

<table>
<thead>
<tr>
<th></th>
<th>Star RandomAccess</th>
<th>Single RandomAccess</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.00655539</td>
<td>0.00793102</td>
</tr>
</tbody>
</table>

Table 5.8: LCG GUPS Star and Single Random Access N=524288

5.2.6 FFT

<table>
<thead>
<tr>
<th></th>
<th>Star FFT</th>
<th>Single FFT</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.0144406</td>
<td>0.0145445</td>
</tr>
</tbody>
</table>

Table 5.9: Star and Single FFT Benchmark with FFT N=131072
5.2.7 Ping-Pong

<table>
<thead>
<tr>
<th>Max Latency</th>
<th>Min. Latency</th>
<th>Max. Bandwidth</th>
<th>Min. Bandwidth</th>
</tr>
</thead>
<tbody>
<tr>
<td>541.24</td>
<td>125.125</td>
<td>0.0815112</td>
<td>0.0116486</td>
</tr>
</tbody>
</table>

Table 5.10: Ping Pong Benchmark (Latency=useconds; Bandwidth=GBytes/s)

5.3 HPC Codes

5.3.1 Jacobi Iterative Solver

The Parallel Image Processing application has been run on a range of core counts on both Tibidabo and HeCToR. Figure 5.3.1 shows the execution of 10000 iteration of the solver on Tibidabo whilst details about the run on HeCToR can be seen in Figure 5.3.1.

![Figure 5.4: 2D Jacobi Solver 10000 Iterations on Tibidabo](image)
Figure 5.5: 2D Jacobi Solver 20000 Iterations on HeCToR

As one can see in the above figures, on both systems the application obtains similar scalability results with the increase in core count. It is important to note however the discrepancies in the number of iterations executed by each system with HeCToR performing 2x more.

Perhaps one of the most interesting results is based on the obtained speedup when increasing the core count for each run (see Figure 5.3.1). The Tibidabo cluster obtains better performance with scaling compared to HeCToR therefore achieving a higher degree of scalability. This is however due to the fact that the ARM CPU’s low performance make for a very balanced system with the latency of the Gigabit Ethernet being adequately low to keep the CPU busy on computing the relevant pixels. On the other hand, the reason for HeCToR’s inferior performance is the reduced size of the image block that each process in the decomposition receives which in turns means that the CPU traverses it too quickly for the Gemini interconnect.
5.3.2 Black Scholes Option Pricing

Running the Black-Scholes Asian Option Pricing application on both Tibidabo and HeCToR on up to 128 processing cores obtained the scalability results illustrated in Figure 5.3.2 and 5.3.2.
Bearing in mind that this particular application is compute bound with almost no communication being present except the distribution of initial stocks and subsequent gathering of results, one can observe how the AMD Opteron 8200 chip is two orders of magnitude faster than the ARM Cortex-A9 chip found in the Tegra 2 SoC. The full figures of the above executions are presented in Table 5.11 for further inspection.
<table>
<thead>
<tr>
<th>No. Cores</th>
<th>Tibidabo</th>
<th>HeCToR</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1201.167</td>
<td>83.265</td>
</tr>
<tr>
<td>2</td>
<td>598.357</td>
<td>43.590</td>
</tr>
<tr>
<td>8</td>
<td>168.907</td>
<td>612.228</td>
</tr>
<tr>
<td>16</td>
<td>97.509</td>
<td>6.552</td>
</tr>
<tr>
<td>32</td>
<td>71.382</td>
<td>4.076</td>
</tr>
<tr>
<td>64</td>
<td>46.832</td>
<td>4.033</td>
</tr>
<tr>
<td>128</td>
<td>43.583</td>
<td>3.873</td>
</tr>
</tbody>
</table>

Table 5.11: Black-Scholes Asian Option Pricing Execution Times in seconds

Figure 5.3.2 illustrates the speedup obtained when scaling the number of processors.

![Black-Scholes Option Pricing Speedup](image)

Figure 5.9: Black-Scholes Asian Option Pricing Speedup

### 5.3.3 Particle Simulation

Figure 5.3.3 and 5.3.3 portray the results obtained by running the Particle Simulation application on up to 128 cores on both Tibidabo and HeCToR.
Figure 5.3.3 presents the speed up obtained with the increase in core count.
Table 5.3.3 contains the execution time registered on each run on both systems.

<table>
<thead>
<tr>
<th>No. Cores</th>
<th>Tibidabo</th>
<th>HeCToR</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1604.274</td>
<td>23.582</td>
</tr>
<tr>
<td>4</td>
<td>847.402</td>
<td>11.894</td>
</tr>
<tr>
<td>8</td>
<td>424.715</td>
<td>6.208</td>
</tr>
<tr>
<td>16</td>
<td>213.263</td>
<td>3.2141</td>
</tr>
<tr>
<td>32</td>
<td>106.618</td>
<td>1.578</td>
</tr>
<tr>
<td>64</td>
<td>53.964</td>
<td>0.935</td>
</tr>
<tr>
<td>128</td>
<td>49.433</td>
<td>0.461</td>
</tr>
</tbody>
</table>

Table 5.12: Particle Simulation Execution Time in seconds

Tables 5.3.3 and 5.3.3 display the amount of time spent in computation and communication.
<table>
<thead>
<tr>
<th>No. Cores</th>
<th>Computation</th>
<th>Communication</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1604.274</td>
<td>0.125</td>
</tr>
<tr>
<td>4</td>
<td>847.109</td>
<td>0.293</td>
</tr>
<tr>
<td>8</td>
<td>423.875</td>
<td>0.839</td>
</tr>
<tr>
<td>16</td>
<td>210.945</td>
<td>2.318</td>
</tr>
<tr>
<td>32</td>
<td>105.534</td>
<td>1.083</td>
</tr>
<tr>
<td>64</td>
<td>53.049</td>
<td>0.918</td>
</tr>
<tr>
<td>128</td>
<td>26.610</td>
<td>22.823</td>
</tr>
</tbody>
</table>

Table 5.13: Computation vs. Communication in seconds on Tibidabo

<table>
<thead>
<tr>
<th>No. Cores</th>
<th>Computation</th>
<th>Communication</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>23.510</td>
<td>0.071</td>
</tr>
<tr>
<td>4</td>
<td>11.687</td>
<td>0.207</td>
</tr>
<tr>
<td>8</td>
<td>6.170</td>
<td>0.029</td>
</tr>
<tr>
<td>16</td>
<td>3.160</td>
<td>0.053</td>
</tr>
<tr>
<td>32</td>
<td>1.537</td>
<td>0.041</td>
</tr>
<tr>
<td>64</td>
<td>0.735</td>
<td>0.199</td>
</tr>
<tr>
<td>128</td>
<td>0.361</td>
<td>0.100</td>
</tr>
</tbody>
</table>

Table 5.14: Computation vs. Communication in seconds on HeCToR

5.4 Summary

This chapter has presented the results obtained from running the HPCC synthetic benchmarks and a number of HPC applications.

The next chapter shall contain the project conclusions.
Chapter 6

Conclusions

This project has presented thorough research into the current and future energy-efficient compute architectures that can be used on the way to Exascale computing.

An analysis of the Barcelona Supercomputing Centre’s Tibidabo cluster has been presented in regards to its configuration and performance through the utilisation of synthetic benchmarks and real HPC codes. Although the cluster is relatively efficient in terms of power consumption averaging approximately 400-450 Watts per rack, the low-powered Tegra 2 processors lack the necessary performance and die density in order to compete with the x86 architectures which can be found in the majority of today’s HPC systems. This balance however can change bearing in mind the projected improvements to the ARM architecture which will most likely lead to a more fierce competition in the HPC ecosystem.
Chapter 7

Further Work

This project can be further built upon by following the initial objectives of asserting the computational performance increase one could obtain through the addition of a massively parallel discrete GPU on the Q7 board therefore complementing the existing ARM chip.

This could be achieved through either the development of a cluster using a platform such as the NVIDIA Carma board or on the next scheduled "Pedraforca" prototype which will contain such addition.
Chapter 8

Project Post-Mortem

A post-mortem of this project is necessary in order to present the numerous changes in objectives and scope that occurred due to external events that were out of the project’s sphere of control.

In the first instance, the project was conceived with the aim of building a cluster out of the recently announced Raspberry Pi device [68]. The aim was to empirically evaluate its usage as a compute node bearing in mind the low-powered ARM architecture it was based upon and small retail price. However, unscheduled delays in the manufacturing process due to various issues and the inability of purchasing more than one device within its first public release forced the project unto a different trajectory.

Subsequently, the focused switched on other available ARM platforms that could be used as building blocks for low-powered high performance computing systems. After reviewing some of these alternatives, the NVIDIA Carma Development Kit made for a very logical choice due to its design principles perfectly matching the requirements set out initially in the project. The problem however was that the platform was still awaiting official release although estimates gathered from both NVIDIA and the manufacturer SECO suggested an expected date surrounding the May-June interval. These estimates seemed to be valid when a prototype of the board was presented at the GPU Technology Conference in May held and organised by NVIDIA.

The adoption of the Carma architecture as the primary investigation focus led to the formal identification of the project’s objectives. These were:

- The development of a compute cluster hosting a number of these devices.
- Investigation into OS fine-tuning on ARM.
- Benchmarking using synthetic tools such as LINPACK.
- Developing, porting and benchmarking a number of scientific applications commonly used in HPC unto the platform.

It was envisaged that the performance metrics gathered from implementing the above could be compared with the Barcelona Supercomputing Centre’s Tibidabo prototype.
cluster which was using a prior version of the Carma architecture also manufactured by SECO but containing an inferior ARM CPU and no discrete GPU. This would have allowed for an empirical evaluation regarding the possible increase in computational efficiency that can be obtained through the synergy of an ARM-based CPU and a low-powered massively parallel discrete GPU such as the NVIDIA Quadro1000m which was included on the Carma kit.

A hardware donation request for two Carma Development kits was submitted to the NVIDIA Hardware Donation program with a further order for 4 of these devices being made to an external reseller. Unfortunately, the Carma’s availability suffered a number of modifications for unknown reasons with the device not being shipped in time with the project’s deadline which inflicted the urgent need for another contingency plan. This resulted in further modification to the project’s plan and arrived at the current incarnation of this endeavour where Tibidabo was used as a replacement for what was intended to be a Carma-based compute cluster. This resulted in the current project form which was presented above.

With hindsight and 20:20 vision, a quicker resolution on applying the elaborated contingency plans should have been made as soon as the first delays in the Carma release occurred therefore not wasting valuable time on GPU-accelerated versions of the codes that were intended for that particular platform. It is perhaps through naivety that one expected the announced release date to be infallible in front of any arising issue even though the project experienced similar misfortunes previously with the Raspberry Pi. As a result, the lesson is to never regard hardware or software release dates as immovable or certain and always pay precious attention to the development of contingency plans and their quick application when required.
Appendix A

Synthetic Benchmarking Results

A.1 Single Node HPL Output

<table>
<thead>
<tr>
<th>T/V</th>
<th>N</th>
<th>NB</th>
<th>P</th>
<th>Q</th>
<th>Time</th>
<th>Gflops</th>
</tr>
</thead>
<tbody>
<tr>
<td>WR00L2L2</td>
<td>10000</td>
<td>112</td>
<td>1</td>
<td>2</td>
<td>567.71</td>
<td>1.175e+00</td>
</tr>
</tbody>
</table>

$\|Ax-b\|_\infty/(\varepsilon*(\|A\|_\infty*\|x\|_\infty+\|b\|_\infty)*N) = 0.0014515 \ldots$ PASSED

Finished 1 tests with the following results:
1 tests completed and passed residual checks,
0 tests completed and failed residual checks,
0 tests skipped because of illegal input values.

End of Tests.
### A.2 Full System HPL Output

<table>
<thead>
<tr>
<th>T/V</th>
<th>N</th>
<th>NB</th>
<th>P</th>
<th>Q</th>
<th>Time</th>
<th>Gflops</th>
</tr>
</thead>
<tbody>
<tr>
<td>WR00L2L2</td>
<td>99687</td>
<td>116</td>
<td>4</td>
<td>53</td>
<td>6394.60</td>
<td>1.033e+02</td>
</tr>
</tbody>
</table>

\[ \|Ax-b\|_\infty/(\text{eps} \times (\|x\|_\infty + \|b\|_\infty)) \times N = 0.0004041 \ldots \]

Passed

Finished 1 tests with the following results:
1 tests completed and passed residual checks,
0 tests completed and failed residual checks,
0 tests skipped because of illegal input values.

End of Tests.
Appendix B

Slurm Job Script

#!/bin/bash

# @ job_name = HPL_mpich2
# @ initialdir = .
# @ output = linpack_%j.out
# @ error = linpack_%j.err
# @ total_tasks = 166
# @ tasks_per_node = 2
# @ cpus_per_task = 1
# @ wall_clock_limit = 18:00:00
# @ power_graph = 1

printenv | grep SLURM_NODELIST | sed 's/SLURM_NODELIST=//g' >> linpack_${SLURM_JOBID}.out

module unload openmpi
module load mpich2

srun xhpl
Appendix C

Wilson and Clover QCD Scaling Benchmark

Figure C.1: QCD Scalability on Tibidabo
Bibliography


[38] Parsons M. CRESTA: Software co-design on the road to exascale [Conf. Session] International Supercomputing Conference 2012.

58


