Literature Review

Michael Kruger

November 6, 2015

1 Introduction

The aim of this project is to create and benchmark a cluster of Parallella boards. In preparation for this, information relevant to the project is reviewed. The topics discussed include high performance computing (HPC), HPC benchmarks, and the Parallella boards themselves.

2 High Performance Computing

HPC is the term for very fast systems aimed at processing large amounts of information quickly. High performance computers are made up of multiple processors as the speed of a single processor has reached its limits due to physics [1]. HPC is most cheaply obtained using cluster computing, as most places needing large amounts of processing power have multiple computers readily available [2]. HPC is used to run high cost simulations that would be too expensive or difficult to do physically; these require large amounts of computational power to be completed in a timely manner [3], [4], and therefore, more powerful machines are continuously being built. HPC has evolved over time with the number of cores in a single computer approaching the millions and performance reaching multiple petaFLOPS (10^{15} floating-point operations per second) [5].

Owing to current size and speed constraints on single core processors, it has been found that running multiple slower cores is both more efficient and faster. Algorithms need to take into account ways in which to split the workload evenly between multiple processors if they want to obtain faster execution speeds using this type of architecture [1]. Many compilers have special flags so that they can optimise programs for parallel computation [6]; this, however, only achieves a minor boost in speed when compared with an efficient algorithm that splits the work into multiple pieces that can be distributed among multiple processors.

According to David Geer [1], to take advantage of multiple cores, programs need to be rewritten so that they can run on multiple threads, with each thread assigned to a separate processor.

2.1 Concepts/Terminology

Throughput The rate at which data can be successfully transferred over a channel.

Shared Memory Memory, over which multiple processes have control and which is shared between them.

Distributed memory This is the term used in a multi-core system when a processor has its own private memory that it can access and use; however, when it needs information from another process, it has to communicate with the other process and request the particular data.

Bottleneck A bottleneck occurs when the effectiveness of a system is restricted by a single or small number of resources.

Latency This refers to the amount of time required for an instruction to travel from its source to its location and be acted upon. A large amount of latency is detrimental as the time to pass information around a system will become a bottleneck and the system will not be able to make use of all its computational power.

FLOPS Floating Point Operations Per Second is the usual measurement of a high performance computer. It refers to the number of instructions using floats that a system can compute per second. It is usually referred to using a prefix such as giga for 10^9 or peta for 10^{15}.

2.2 Different Architectures

Clusters With the demand for large amounts of processing power, various ways of creating supercomputers cheaply have appeared. Clusters of computers connected on a network can be purposed to work together as a supercomputer. With the increased speed and decreased latency of the Internet, it is possible to create a cluster using computers from all over the world; this has led to programs and applications that allow a computer to connect to a pool of other computers and add its processing power to the computation. There are, however, some factors limiting the effectiveness of cluster computing. These

include building a switch to keep up with the speed of a single core processor and creating compilers that make good use of multiple processors. There are two generally used methods for controlling communication within a cluster:

MPI The individual nodes of the cluster can communicate with each other using a message passing interface (MPI), which provides a thread safe application programming interface (API) that allows the work to be effectively delegated to multiple nodes on the network [7], [8] and information passed between each node so that it can be worked on. More information on MPI is provided in Section **??** with the overview of MPICH.

Parallel Virtual Machine uses a parallel virtual machine (PVM) approach, which combines all the nodes and allows them to appear as a single PVM. This PVM handles all the message passing, task scheduling, and data conversions. To set this up, each node of the cluster needs the same PVM image installed and must be marked as a PVM node. Parallel virtual machines are popular due to the ease with which the cluster can be managed[9]. Some of the available PVMs are reviewed in Section **??**.

2.3 Existing HPC Architectures

This subsection gives an overview of some existing high performance computing systems.

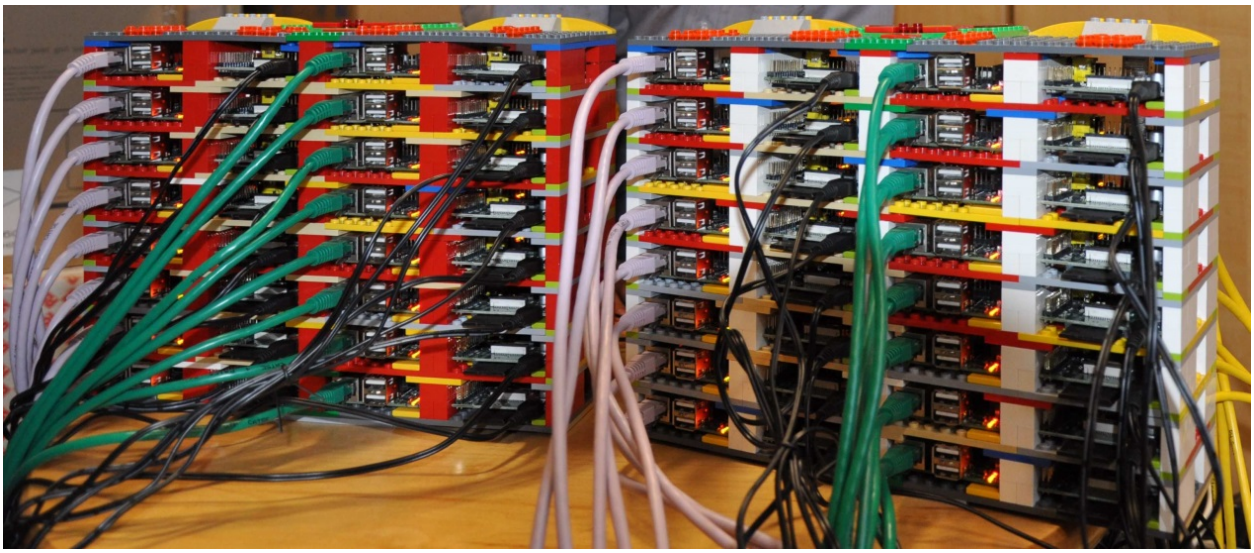

Figure 1: Photo of the Iridis-pi at the University of Southampton [10]

Iridis-pi The Iridis-pi is a cluster constructed from 64 Raspberry Pi model B nodes and held together with Lego. The Iridis-Pi was created by a team at the University of Southampton. Each Raspberry Pi has a 700 MHz ARM processor, 256 MB of RAM, a 16GB SD card, and a 100Mb/s network interface. The benefits of the cluster are its cheap price, compact size, and low power consumption. Using the HPL benchmark and all 64 nodes the Iridis-pi achieves a throughput of 1.14 gigaFLOPS[10]. The price of a Raspberry Pi model B at time of writing is 450 South African rand [1] bringing the cost of the 64 Iridis-pi nodes to 28800 South African rand.

Figure 2: Photo of the Tianhe-2

Tianhe-2 (Milkyway-2) The most powerful supercomputer according to the Top500 list of the world's supercomputers in November 2015 is the Tianhe-2 (Milkyway-2). On the LINPACK benchmark, it achieved a performance of 33,862.7 teraFLOPS with a theoretical peak of 54,902.4 teraFLOPS[5]. The Tianhe-2 is sixth according to the graph500 November 2015 results. The graph500 uses different benchmarks that are

[1] `http://pifactory.dedicated.co.za/product/raspberry-pi-1-model-b/`

focused on data intensive supercomputer applications. The Tianhe-2 performed at 2061.48 GTEPS (10^9 traversed edges per second)[11]. Developed by China's National University of Defence Technology (NUDT) in collaboration with the Chinese IT firm Inspur, the Tianhe has managed to maintain the top spot since June 2013. The Tianhe is composed of 16,000 computer nodes, each consisting of two Intel Xeon IvyBridge processors and three Xeon Phi processors with access to 88 GB of RAM; this equates to 3.12 million cores and 1.404 petabytes of RAM [2]. Running NUDT's own operating system Kylin Linux, which is optimised for high-performance parallel computing as well as having support for power management and high-performance virtual computing zone, the Tianhe-2 uses MPICH2 with a customised GLEX channel[5] for executing HPC programs. A picture of a portion of the supercomputer can be seen in Figure **??** [3]

2.4 Benchmarks for HPC

The HPC Challenge Benchmark is a set of seven benchmarks for testing an HPC system's ability to cope with different scenarios, thereby giving an indication of real-world performance [12]. These benchmarks include:

HPL "A Portable Implementation of the High-Performance LINPACK Benchmark for Distributed-Memory Computers" The computers in the Top500 list are ranked by the HPL NxN benchmark results[5]. The reason for choosing LINPACK as a benchmark is that it is widely used, and the benchmarked performance of a large number of systems is available. LINPACK is a collection of Fortran subroutines for solving systems of linear equations. Owing to the distributed nature of both the memory and computing nodes of HPC, the highly-parallel LINPACK (HPL) benchmark was created to compare results on these systems. In the Top500 list, the results taken into account are:

- R_{max} – the maximum performance achieved by LINPACK
- N_{max} – how large the problem was to get the result in R_{max}
- $N_{1/2}$ – the size of the problem to get half of R_{max}
- R_{peak} – theoretical peak performance.

[2] http://www.extremetech.com/computing/159465-chinas-tianhe-2-supercomputer-twice-as-fast-as-does-titan-shocks-the-world-by-arriving-two-years-early

[3] Picture taken from : `http://chinadaily.com.cn/business/tech/img/attachement/jpg/site1/20140626/eca86bd9e2eb1516011b02.jpg`

all of which are taken from the HPL benchmark [5], [13]–[16].

DGEMM is a method that calculates the product of double precision matrices and measuring the rate of execution, provides insight into the performance of the HPC device.

STREAM provides a measurement of the "sustainable memory bandwidth and the corresponding computation rate for simple vector kernels"[17].

PTRANS (parallel matrix transpose) forces pairs of processors to communicate simultaneously, testing the communication capacity of the network [4].

Random Access By providing a measurement in GUPS (giga updates per second), this benchmark measures random memory access [5].

FFT (Fast Fourier Transform) This benchmark measures "the floating point rate of execution of double precision complex one-dimensional Discrete Fourier Transform (DFT)"[12], [18].

Communication bandwidth and latency Using the effective bandwidth benchmark [6], latency and bandwidth are measured for a number of simultaneous communication patterns[12].

All these benchmarks measure different aspects that are useful in the real world and provide an idea of what should be tested.

3 Software For Clusters

Since we are deploying a cluster, in this section we only discuss software relevant to cluster computers.

3.1 MPICH

MPICH is a high performance, portable and widely used implementation of the Message Passing Interface (MPI) standard. MPICH was created for distributed memory systems, with the idea of portability and high performance in mind. Excellent results have been

[4] `http://www.netlib.org/parkbench/html/matrix-kernels.html`
[5] http://icl.cs.utk.edu/projectsfiles/hpcc/RandomAccess/
[6] `https://fs.hlrs.de/projects/par/mpi//b_eff/`

achieved with MPICH, which is the most used implementation of MPI in the world, and its derivatives. MPICH is able to work in many different environments and take advantage of what is available to increase performance while maintaining portability, for example, using shared memory to pass messages between processors faster. MPICH is distributed as source code and has an open-source freely available licence [8], [19], [20].

3.2 Open-MPI

Open-MPI is an open-source implementation of the Message Passing Interface [7]. It has multiple partners from across the HPC community, maintaining its libraries[41]. These partners include ARM, which provided the Zync chip for use on the Parallella board[8]. Open-MPI conforms fully to MPI-3.1 standards, supports multiple operating systems, and is provided by default on the Parallella board Ubuntu distribution.

3.3 OpenMP

OpenMP is "an industry standard API for shared-memory programming" [21]. A shared-memory parallel system describes a multi-processor system where individual processors share one memory location [22]. Each processor can still have its own personal cache memory to work with as the speed difference between main memory and processor memory would cripple the speed if the processor needed to pick up everything from the shared memory space. OpenMP was introduced to fix the inability of compilers to make good decisions on how to split up a program to take advantage of multiple processors; although this is possible for simpler programs, a user would need to cast a more discerning eye for more complex problems [22]. OpenMP provides an alternative to message passing in parallel programming. OpenMP is a set of routines and compiler directives to manage shared-memory parallelism. The OpenMP standard is made up of four parts, namely, control structure, data environment, synchronisation, and runtime library [21], which can be added to a sequential program written in C, C++ or Fortran [22].

[7] http://www.open-mpi.org/
[8] http://www.open-mpi.org/about/members/

3.4 Rocks Cluster Distribution

Rocks is a Linux distribution, with its latest version Sidewinder based on Cent-OS 6.6 [1], but only available for 64-bit machines. It is unlikely that Rocks will be used in the construction on the Parallella cluster as there does not seem to be an official release for the ARM processor, which is used by the Parallella. Rocks is an open-source distribution that was built to provide a simple environment in which to build computational clusters, grid endpoints and visualisation tiled-display walls [1].

4 Parallella

The Parallella board is an "affordable, energy efficient, high performance, credit card sized computer"[23] that aims to provide a platform for developing and implementing high performance parallel processing. The 66-core version of the Parallella board achieves over 90 gigaFLOPS (10^9 floating point operations per second), while the 16-core version can reach 32 gigaFLOPS using only about 5 Watts. The Parallella has a 1-Gbps Ethernet port allowing a large amount of information to be passed quickly over the network. This increases its ability to work in a cluster as it can pass information to its peers rapidly, provided that the switch is capable of handling the 1Gbps bandwidth.

The aim of creating the Parallella board was to make parallel computing more accessible by creating an affordable, open-source, and open-access platform.

The price of a Parallella board starts at $99 (at the time of writing) for the 16-core board and uses a customised ARM implementation of Linux (Ubuntu 14.04). The Parallella is new and software that takes advantage of this is still being developed.[24], [25]

Programming for the Epiphany chip is done in C and the Parallella team have provided some basic primitives with the SDK (Software Development Kit). Memory addressing, barriers, and communication between eCores are a few examples of what is provided by the SDK.

To run programs on the Epiphany chip, a workgroup of cores needs to be set up. This can be done using the provided SDK to give a starting node and the number of columns and rows in the matrix of cores [7], [26]–[28].

[1] http://www.rocksclusters.org/

4.1 Specifications

As the proposed cluster will make use of the 16-core version of the Parallella board, the technical specifications of this board are given below [25].

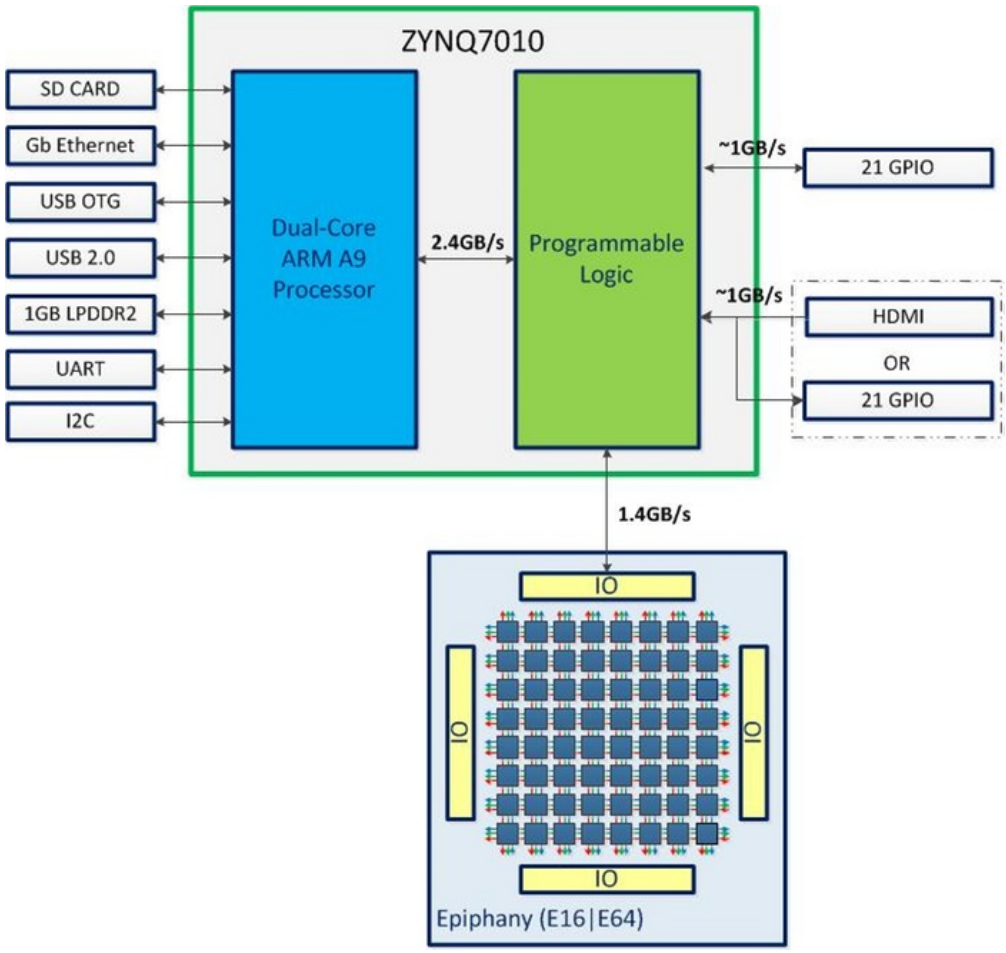

Figure 3: Architectural layout of the Parallella board (taken from [23])

Figure **??** shows that the interfaces and operating system are run with the dual-core ARM processors; the programs running on the ARM processors can use the Epiphany libraries provided with the SDK to set up and run programs on the individual cores [25], [29][26], [28].

The Parallella makes use of a Zynq-Z7010 dual-core ARM A9 CPU to run the operating system and programs not designed to run on the Epiphany chip. A rundown of the Parallella components:

- Zynq-Z7010 Dual-core ARM A9 CPU

- 16-core Epiphany Co-processor
- 1GB DDR3 RAM
- MicroSD Card: Allows storage of local files.
- USB 2.0
- Up to 48 GPIO signal
- Gigabit Ethernet: The high-speed Gigabit Ethernet allows for rapid transfer of data across a network allowing the cluster to communicate with lower latency.
- HDMI port
- Linux Operating System: The Linux operating system is well supported by multiple MPI libraries [8], [19], [20], [41][9].
- 54mm x 87mm form factor: The small form factor of each of the boards makes it highly portable even if using multiple boards.

Figure **??** illustrates the 2D array of cores, which Adapteva calls eCores. Each eCore has a 1GHz RISC CPU, 32 KB of local memory, a network interface, and a direct memory access (DMA) engine. This matrix is connected to the rest of the chip via a router. The router communicates with the rest of the chip via three connections: the blue connector is the on-chip write network, green is the off-chip write network, and red is the read request network. The eCore CPU is super-scalar and can execute two floating-point operations and a 64-bit memory load/store operation in every clock cycle. The local memory can provide up to 32 bytes per clock cycle of bandwidth [25], [29][26], [28].

The Epiphany processor on this board is the Epiphany III (E16G301), the feature summary of which is given below [26]:

- 16 high performance RISC CPU cores:
- C/C++ and OpenCL programmable
- 32-bit IEEE floating point support

[9] `http://www.open-mpi.org/`

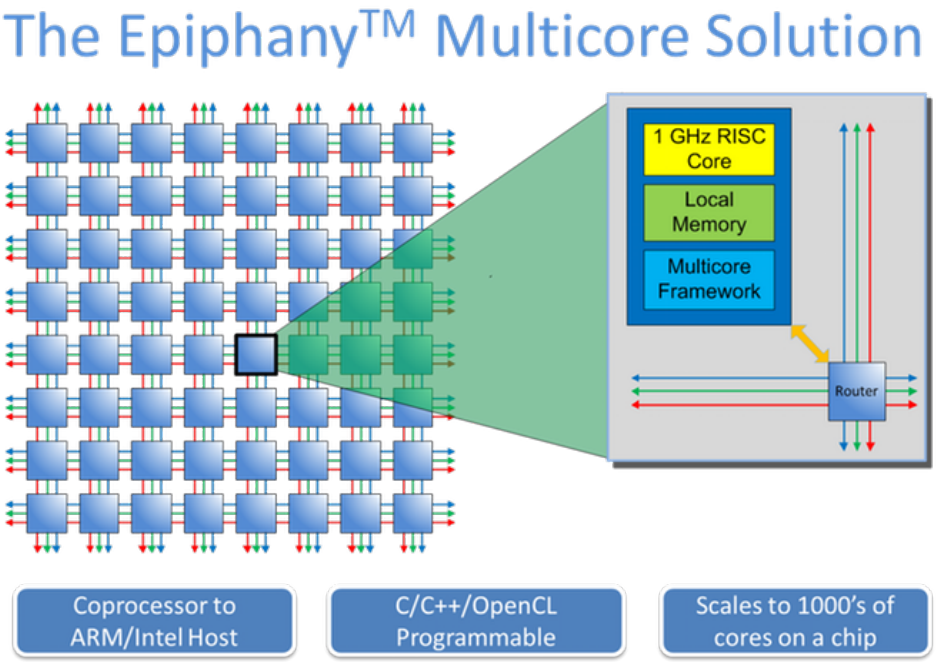

Figure 4: Epiphany mesh architecture

- 512KB on-chip distributed shared memory: This can be used to pass messages and share data across the chip.
- 32 independent DMA channels
- Up to 1GHz operating frequency
- 32 gigaFLOPS peak performance
- 512 GB/s local memory bandwidth: The access speed of each RISC core's local RAM.
- 64 GB/s Network-On-Chip bisection bandwidth: The speed at which message passing takes place on the chip.
- 8 GB/s off-chip bandwidth: The performance of communicating off chip.
- 1.5ns network per-hop latency
- <2 Watt maximum chip power consumption: The power consumption of the Epiphany processor; combined with the rest of the board, the total is 5 W.

4.2 Existing Parallella Configurations

Below we discuss a few of the HPC configurations that have been implemented using Parallella boards.

Parallac The Parallac is a cluster consisting of eight Parallella boards for the main computation and two intel NUCs (next unit of computing) that are each equipped with an Intel i3 processor, 16 GB of RAM and 120 GB of SSD storage. The design of the configuration was inspired by the Cray 1 supercomputer. The Parallellas are powered using their mounting holes connected to a power supply unit (PSU) via a powered plate of copper. The configuration is compact and cables are well managed[30]. Unfortunately, the Parallac website does not show any results on how well this system performs on any benchmarks.

supercomputer.io Supercomputer.io is a project to create a community-hosted supercomputer created by connecting the Parallellas of anyone who signs up and sets up their Parallella with the required software and hardware. These Parallellas are connected to each other and the supercomputer.io network over the Internet. Scientists can then submit requests to use the cluster for their research[31]. The supercomputer.io project intends supporting other boards in time but at the time of writing it only provides support for Parallella boards. To connect a Parallella board to the supercomputer.io network, the Parallella must have at least a 4GB SD card and a connection to the Internet. The supercomputer.io operating system is copied onto the SD card, and using an active Internet connection, the Parallella connects to the supercomputer.io network and waits for work to be allocated to it [31].

5 Summary

The three main topics researched in this chapter were HPC, doftware options for clusters and an overview of the Parallella board. Aspects of HPC that we discussed included how the performance of an HPC system is measured and what the different types of high performance configurations are. The cluster software looked at included message passing implementations, shared memory and a cluster suite Rocks. Finally, the Parallella board was investigated and information on existing Parallella cluster implementations were investigated. Of the cluster configurations found, no benchmark results where obtained.

References

[1] D. Geer, "Chip makers turn to multi-core processors," *Computer*, vol. 38, no. 5, pp. 11–13, 2005.

[2] R. Buyya, "High performance cluster computing," *New Jersey: Prentice Hall*, 1999.

[3] T Tezduyar, S Aliabadi, M Behr, A Johnson, V Kalro, and M Litke, "Flow simulation and high performance computing," *Computational Mechanics*, vol. 18, no. 6, pp. 397–412, 1996, test.

[4] K. Sanbonmatsu and C.-S. Tung, "High performance computing in biology: Multimillion atom simulations of nanoscale systems," *Journal of structural biology*, vol. 157, no. 3, pp. 470–480, 2007.

[5] (). Top 500 super computers. Accessed: 2015.02.25, [Online]. Available: `http://www.top500.org/`.

[6] D. A. Padua and M. J. Wolfe, "Advanced compiler optimizations for supercomputers," *Communications of the ACM*, vol. 29, no. 12, pp. 1184–1201, 1986.

[7] D. Richie, J. Ross, S. Park, and D. Shires, "Threaded MPI programming model for the epiphany risc array processor," *Journal of Computational Science*, 2015.

[8] W. Gropp, E. Lusk, N. Doss, and A. Skjellum, "A high-performance, portable implementation of the MPI message passing interface standard," *Parallel computing*, vol. 22, no. 6, pp. 789–828, 1996.

[9] A. Geist, *PVM: Parallel virtual machine: A users' guide and tutorial for networked parallel computing*. MIT press, 1994.

[10] S. Cox, J. Cox, R. Boardman, S. Johnston, M. Scott, and N. OBrien, "Iridis-pi: A low-cost, compact demonstration cluster," English, *Cluster Computing*, vol. 17, no. 2, pp. 349–358, 2014, ISSN: 1386-7857. DOI: `10.1007/s10586-013-0282-7`. [Online]. Available: `http://dx.doi.org/10.1007/s10586-013-0282-7`.

[11] (). Graph 500. Accessed: 2015.05.27, [Online]. Available: `http://www.graph500.org/`.

[12] I. C. Laboratory. (2015). HPC challenge benchmark. Accessed: 2015.05.29, [Online]. Available: `http://icl.cs.utk.edu/hpcc/`.

[13] T. Davies, C. Karlsson, H. Liu, C. Ding, and Z. Chen, "High performance linpack benchmark: A fault tolerant implementation without checkpointing," in *Proceedings of the international conference on Supercomputing*, ACM, 2011, pp. 162–171.

[14] J. Dongarra and P. Luszczek, "Linpack benchmark," in *Encyclopedia of Parallel Computing*, Springer, 2011, pp. 1033–1036.

[15] J. D. A. Petitet R. C. Whaley and A. Cleary. (). HPL - a portable implementation of the high-performance linpack benchmark for distributed-memory computers.

[16] A. P. Jack J. Dongarray Piotr Luszczeky. (Dec. 2001). The LINPACK benchmark: Past, present, and future. Accessed: 2015.05.28, [Online]. Available: `http://www.netlib.org/utk/people/JackDongarra/PAPERS/hpl.pdf`.

[17] J. D. McCalpin. (). Stream: Sustainable memory bandwidth in high performance computers. Accessed: 2015.05.29, [Online]. Available: `http://www.cs.virginia.edu/stream/`.

[18] D. Takahashi. (). Ffte: A fast fourier transform package. Accessed: 2015.05.29, [Online]. Available: `http://www.ffte.jp/`.

[19] P. Bridges, N. Doss, W. Gropp, E. Karrels, E. Lusk, and A. Skjellum, "User's guide to MPICH, a portable implementation of MPI," *Argonne National Laboratory*, vol. 9700, pp. 60 439–4801, 1995.

[20] W. Gropp and E. Lusk, "Installation guide for mpich, a portable implementation of MPI," Technical Report ANL-96/5, Argonne National Laboratory, Tech. Rep., 1996.

[21] L. Dagum and R. Menon, "Openmp: An industry standard api for shared-memory programming," *Computational Science & Engineering, IEEE*, vol. 5, no. 1, pp. 46–55, 1998.

[22] B. Chapman, G. Jost, and R. Van Der Pas, *Using OpenMP: Portable shared memory parallel programming*. MIT press, 2008, vol. 10.

[23] (). The parallella board. Accessed: 2015-03-01, [Online]. Available: `https://www.parallella.org/`.

[24] (). Parrallella kickstarter. Accessed: 2015.05.05, [Online]. Available: `https://www.kickstarter.com/projects/adapteva/parallella-a-supercomputer-for-everyone`.

[25] papallella. (). Parallella reference manual. Accessed: 2015.05.05, [Online]. Available: `http://www.parallella.org/docs/parallella_manual.pdf`.

[26] Adapteva. (). Epiphany datasheet. Accessed: 2015.05.05, [Online]. Available: `http://adapteva.com/docs/e16g301_datasheet.pdf`.

[27] A. Olofsson, T. Nordström, and Z. Ul-Abdin, "Kickstarting high-performance energy-efficient manycore architectures with epiphany," *ArXiv preprint arXiv:1412.5538*, 2014.

[28] A. Varghese, B. Edwards, G. Mitra, and A. P. Rendell, "Programming the adapteva epiphany 64-core network-on-chip coprocessor," in *Parallel & Distributed Processing Symposium Workshops (IPDPSW), 2014 IEEE International*, IEEE, 2014, pp. 984–992.

[29] D. Richie. (). Threaded MPI for the epiphany architecture. Accessed: 2015.05.13, [Online]. Available: `https://www.parallella.org/2015/04/09/threaded-mpi-for-the-epiphany-architecture/`.

[30] (). Building the parallac. Accessed: 2015.05.05, [Online]. Available: `http://www.parallac.org/`.

[31] (). Supercomputer.io. Accessed: 2015.05.27, [Online]. Available: `http://supercomputer.io/`.

[32] G. S. Almasi and A. Gottlieb, "Highly parallel computing," 1988.

[33] M. Bakery and R. Buyyaz, "Cluster computing at a glance," *High Performance Cluster Computing: Architectures and Systems*, vol. 1, pp. 3–47, 1999.

[34] J. Dursi. (Apr. 2015). HPC is dying, and MPI is killing it. Accessed: 2015.04.19, [Online]. Available: `http://www.dursi.ca/hpc-is-dying-and-mpi-is-killing-it/`.

[35] P. Luszczek, J. J. Dongarra, D. Koester, R. Rabenseifner, B. Lucas, J. Kepner, J. McCalpin, D. Bailey, and D. Takahashi, "Introduction to the HPC challenge benchmark suite," *Lawrence Berkeley National Laboratory*, 2005.

[36] P. R. Luszczek, D. H. Bailey, J. J. Dongarra, J. Kepner, R. F. Lucas, R. Rabenseifner, and D. Takahashi, "The HPC challenge (HPCC) benchmark suite," in *Proceedings of the 2006 ACM/IEEE conference on Supercomputing*, Citeseer, 2006, p. 213.

[37] M. F. Mergen, V. Uhlig, O. Krieger, and J. Xenidis, "Virtualization for high-performance computing," *ACM SIGOPS Operating Systems Review*, vol. 40, no. 2, pp. 8–11, 2006.

[38] Parallella. (). Parallella-examples. Accessed: 2015.05.06, [Online]. Available: `https://github.com/parallella/parallella-examples`.

[39] J. E. D. D. S. D. J. B. Udaya A. Ranawake Charles V. Packer and T. Sterling, "Beowulf: A parallel workstation for scientific computation," [Online]. Available: `http://www.phy.duke.edu/~rgb/brahma/Resources/beowulf/papers/ICPP95/icpp95.html`.

[40] (). Networkboot parallella board. Accessed: 2015-02-27, [Online]. Available: https://plus.google.com/+shogunx/posts/K5taMhPKurp.

[41] E. Gabriel, G. E. Fagg, G. Bosilca, T. Angskun, J. J. Dongarra, J. M. Squyres, V. Sahay, P. Kambadur, B. Barrett, A. Lumsdaine, *et al.*, "Open MPI: Goals, concept, and design of a next generation MPI implementation," in *Recent Advances in Parallel Virtual Machine and Message Passing Interface*, Springer, 2004, pp. 97–104.

[42] Adapteva. (). Epiphany SDK reference. Accessed: 2015.11.01, [Online]. Available: http://adapteva.com/docs/epiphany_sdk_ref.pdf.