Performance Evaluation of Multi-Core Multi-Cluster Architecture (MCMCA)

Norhazlina Hamid, Robert J. Walters and Gary B. Wills

School of Electronics & Computer Science, University of Southampton,
SO17 1BJ, Southampton, U.K.
{nh3g11, rjw1, gbw}@ecs.soton.ac.uk

Abstract. A multi-core cluster is a cluster composed of numbers of nodes where each node has a number of processors, each with more than one core within each single chip. Cluster nodes are connected via an interconnection network. Multi-cored processors are able to achieve higher performance without driving up power consumption and heat, which is the main concern in a single-core processor. A general problem in the network arises from the fact that multiple messages can be in transit at the same time on the same network links. This paper considers the communication latencies of a multi-core multi-cluster architecture will be investigated using simulation experiments and measurements under various working conditions.

1 Introduction

In the past, it was a trend to increase a processor's speed to get better performance. Moore's Law, which states that the number of transistors on a processor will double approximately every two years has been proven to be consistent due to the transistors getting smaller in successive processor technologies [1]. However, reducing the transistor size and increasing clock speeds causes transistors to consume more power and generate more heat [2]. These concerns limit cost-effective increases in performance which can be achieved by raising processor speed alone. These issues gave computer engineers the idea of designing the multi-core processor, a single processor with two or more cores [3].

Multi-core processors are used extensively in parallel and cluster computing. As far back as 2009, more than 95% of the systems listed in the Top 500 supercomputer list used dual-core or quad-core processors [4]. The motivation in this realm is the advances in multi-core processor technology that makes them an excellent choice to use in cluster architecture [5]. From the combination of cluster computing and multi-core processor, the multi-core cluster architecture has emerged. The multi-core cluster architecture becomes more powerful due to the combination of faster processors and faster interconnection [6].

Overall performance of multi-core cluster always determined by the efficiency of its communication and interconnection networks [7]. Hence, performance analysis of the interconnection networks is vital. A general problem in the network may arise from multiple messages being in transmission at once on the same network links and this will cause delays. Such delays increase the communication latency of the interconnection network and it is therefore important to minimise this. A high communication latency of interconnection network can dramatically reduce the efficiency of the cluster system [8].

Many studies [9-11] have been carried out to improve the performance of multi-core cluster but few clearly distinguish the key issue of the performance of interconnection networks. Although the cluster interconnection network is critical for delivering efficient performance, as it needs to handle the networking requirements of each processor core [8], existing models do not address the potential performance issues of the interconnection networks within multi-core clusters.

Scalability is a very important aspect to examine when evaluating clusters. Abdelgadir, Pathan, & Ahmed [12] find that having good network bandwidth and faster network will produce better scalability of clusters. Scaling up by adding more processors to each node to increase the processing power creates too much heat [3]. The conventional approach to improving cluster throughput is to add more processors but there is a limit to the scalability of this approach; the infrastructure cannot provide effective memory access to unlimited numbers of processors and the interconnection network(s) become saturated [13]. Technological advances have made it viable to overcome these problems by combining multiple clusters of heterogeneous networked resources into what is known as a multi-cluster architecture [14].

We describe a scalable approach to building heterogeneous multi-cluster architecture and are the first investigation into network latency within such architecture.

2 The Architecture

Multi-core processor is a single processor within a chip with two or more cores [3]. Multi-core processors are the answer for the deficiencies of single-core processors; as processor speed increase, the amount of heat produced and the amount of power consume by the processor increase with it. Multi-core processors can perform more work within a given amount of time by dividing the work between two or more cores while reduce the power consumption and dissipate the heat [15]. Due to their greater computing power and cost-to-performance effectiveness, cluster computing uses multi-core processors extensively [16].

A multi-core cluster is a cluster where all the nodes in the cluster have multi-core processors. In addition, each node may have multiple processors (each of which contains multiple cores). With such cluster nodes, the processors in the node share both memory and their connections to the outside. A new architecture known as the Multi-Core Multi-Cluster Architecture (MCMCA) is introduced in **Fig. 1**. The structure of MCMCA is derived from a Multi-Stage Clustering System (MSCS) [13] which is based on a basic cluster using single-core nodes. The MCMCA is built up of numbers of clusters where each cluster is composed of numbers of nodes. The numbers of nodes are determined at run time. Each node of a multi-core cluster has more than one processor. Cores on the same chip share local memory but have their own L2 cache. The interconnection network connects the cluster nodes.

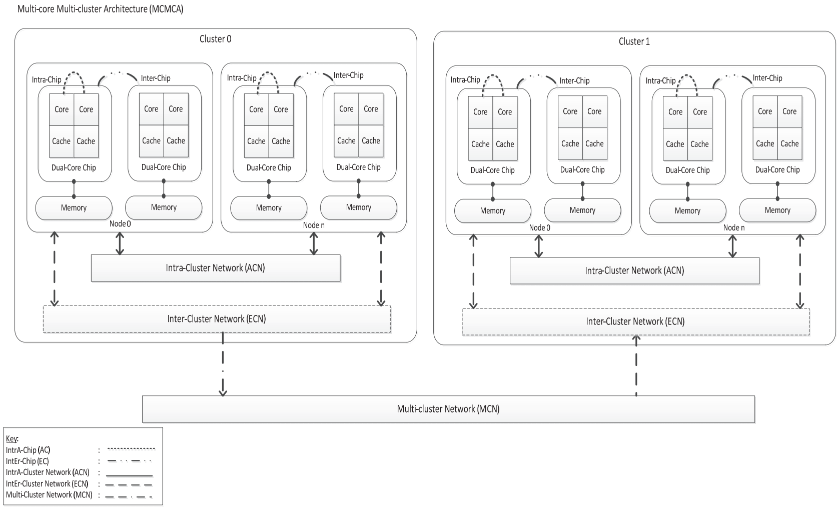
Multi-core Multi-cluster Architecture (MCMCA)
Cluster 0
Cluster 1
Intra-Chip
Inter-Chip
Core
Cache
Dual-Core Chip
Memory
Node 0
Node n
Intra-Cluster Network (ACN)
Inter-Cluster Network (ECN)
Multi-cluster Network (MCN)
Key:
IntrA-Chip (AC) :
IntEr-Chip (EC) :
IntrA-Cluster Network (ACN) :
IntEr-Cluster Network (ECN) :
Multi-Cluster Network (MCN) :

Fig. 1. Overview of the proposed Multi-Core Multi-Cluster Architecture (MCMCA).

3 Communication Network

The performance of a cluster system depends on its communication latency of the interconnection network. The research conjecture is that low communication latency is essential to achieving a faster network and increasing the efficiency of a cluster. In order to understand the communication network of the Multi-Core Multi-Cluster Architecture (MCMCA), this section explains in detail the different types of communication networks.

There are five communication networks in MCMCA. Three of them are commonly found in any multi-core cluster architecture, these are: the intra-chip communication network (AC); the inter-chip communication network (EC) and the intra-cluster network (ACN). The new communication networks introduced in this paper are the intercluster network (ECN) and the multi-cluster network (MCN).

The communication between two processor cores on the same chip is the intra-chip communication network (AC), as shown in **Fig. 2**. Messages from source A to destination B travel via the AC communication network, which acts as a connector between two processor cores on the same chip.

Fig. 3 shows an inter-chip communication network (EC) for communicating across processors in different chips but within a node. Messages travelling to different chips from source A in the same node first have to communicate within the chip via the intra-chip communication network (AC), and then travel between the chips via the EC network to reach their destination B. Each node has two communication connections which are intra-cluster network (ACN) for transmission within a cluster and inter-cluster network (ECN) for transmission between clusters.

An intra-cluster network (ACN) is used for messages within a cluster. In order for messages to cross the nodes, messages have to communicate with the intra-chip communication network (AC) and the inter-chip communication network (EC) to pass between chips. Then messages travel via the ACN to enter different nodes to reach their destination, as shown in **Fig. 4**.

Messages travelling from source A to destination B between clusters communicate via two communication networks to reach other clusters, as shown in **Fig. 5**. An inter-cluster network (ECN) is used to transmit messages between clusters. The clusters are connected to each other via the multi cluster network (MCN). When the messages reach the other cluster, they have to communicate with the ECN of the target cluster before arriving at their destination.

All levels of communication are critical in order to optimise the overall performance of the Multi-Core Multi-Cluster Architecture (MCMCA). The overall communication latency gathered from all communication networks will be calculated. The derived simulation results will be analysed for comparison between the existing architecture and the MCMCA architecture.

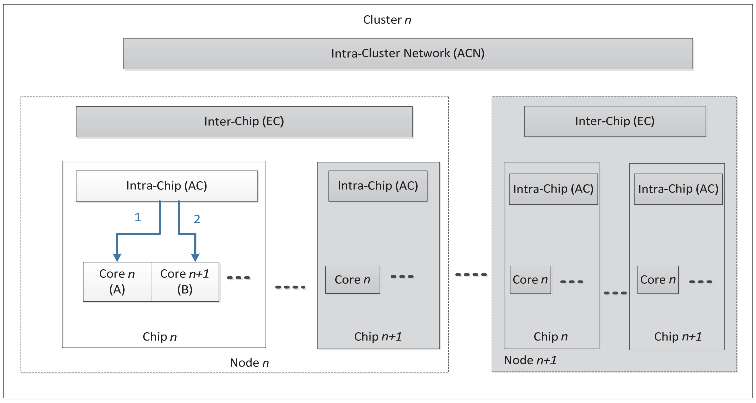

Fig. 2. Communication network flow A for message passing between two processor cores on the same chip.

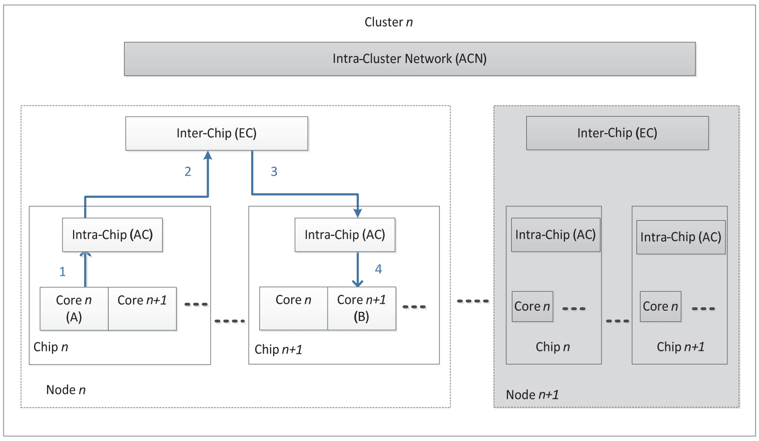

Fig. 3. Communication network flow B for message passing across processors in different chips but within a node.

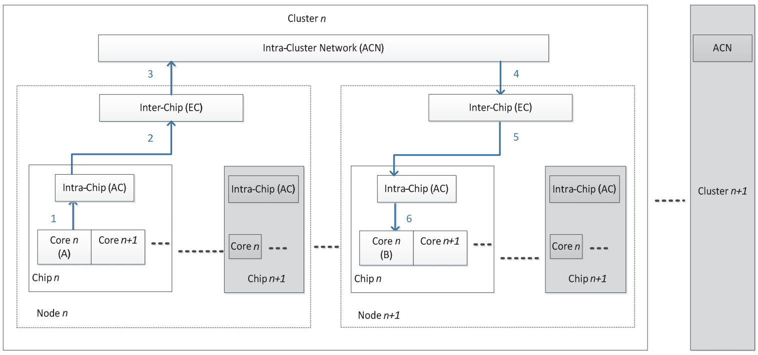

Fig. 4. Communication network flow C for message passing between processors on different nodes but within the same cluster.

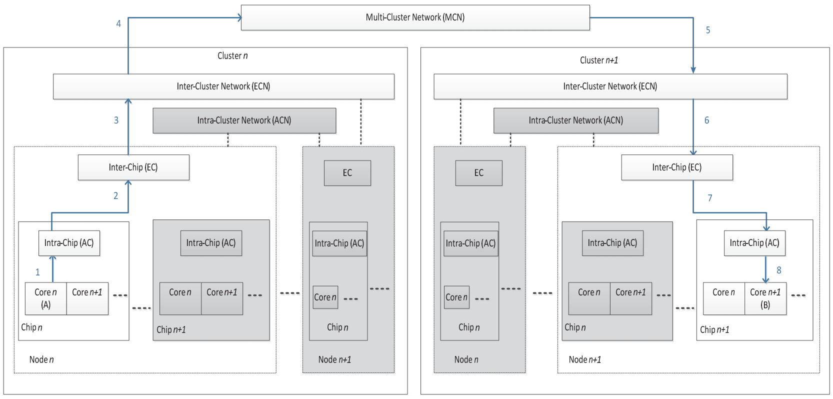

Fig. 5. Communication network flow D for transmitting messages between clusters.

4 Research Methodology

This work will use computer simulation to model the architecture. The model will be validated using computer simulation experiments.

4.1 Simulation Tool

OMNeT++ network simulation tool [17], a C++ based open-source discrete-event simulator has been chosen to model MCMCA. Comparison studies of network simulators have been conducted which involved OMNeT++, MATLAB, ns-2, ns-3, OPNet and QualNet [18-21]. NS-2 is still the most widely used network simulator in academia, but OMNeT++ provides more infrastructures. OMNeT++ is also popular in academia and industry because of its extensibility since it is also open source. There is also plentiful online documentation and mailing lists for general discussion. Although NS-3 demonstrated the best overall performance, NS-3 still needs to improve its simulation credibility [19] and OMNeT++ can be considered as a viable alternative. OPNet has similar foundations to OMNeT++ although it is a commercial product, and contains an extensive model library and provides several additional programs and GUI tools. The other two commercial products are QualNet, which emphasises wireless simulations, and MATLAB, which needs several prerequisite components for its files to function normally.

4.2 Simulation Development

An early stage of simulation experiments under various configurations and design parameters has been completed. The performance evaluation focused on communication latency in the MCMCA architecture. As a preliminary study, the communication network performance experiments are based on a single-core multi-cluster architecture. A simulation model has been built to measure the performance of single-core multi-cluster architecture. This evaluation was then compared to the model of multi-cluster architecture presented by Javadi et al., [22] with the given configuration and parameters to match the work in their papers.

This work focuses on measuring steady-state performance of a network; the performance of a network with a stationary traffic source after it has reached steadiness. A network has reached steadiness when its average queue lengths have reached their steady-state values. To measure steady-state performance, the simulation experiments were conducted in three phases: warm-up, measurement and drain [7]. The network has necessarily reached a steady-state once the network is warmed up [22]. This means that the statistics of the network are stationary and no longer changing with time, which will determine an accurate estimation.

4.3 Simulation Setup

The model behaviour built into each Network Description (NED) file will be captured in C++ files as code and can be edited in the Integrated Development Environment [23]. Each NED file has its own C++ simple module source. Unlike many formats of deterministic discrete event simulation, the model is built at run-time to form a topology that represents the geometric structure and the communication links between the modules. At the start of each execution, the simulator reads the initialisation file (.INI file) that tells the tool which network file is to be simulated. In this initialisation file the parameters of the model, such as number of cores per node, number of clusters, number of messages to be generated, message length (M), flit length (F) and inter-arrival time, are specified. The simulation can also behave with different initialisation inputs and all the values can be stored in an .INI file, usually called omnetpp.ini, containing settings that control how the simulation is executed.

When the simulation is started, it will show how the messages hop from module to module following the routing algorithm. Each message will be partitioned into a sequence of packets first before being generated at each tree-node, following the assumptions that the message destinations are uniformly distributed by using a uniform random number generator. Packets travelling from source to its destination will access the available processor through a chip first, where a chip can contain one or more processors. Then, each processor will divide the packets into the number of cores. If first processor is busy, it will pass the packets to another processor in the same chip within the same node first, before determining via the communication network if other processors in other chips can process the packets. Packets will access the processors, chips and nodes in the same cluster first before accessing other clusters via communication network.

The routing file will determine the path the packets will follow in the network from the source to the destination. Based on the path given, the packets will travel using communication switch to get through the communication network. If there is a situation where more than one packet needs to use the same route, the communication switch will determine which packet can go through first or if the packet needs to queue (buffer) until the route is available. Each packet is time-stamped after its generation and the message completion time is defined on each tree-node to compute message latency. Statistics will be gathered for every event in the simulation for analysis of the results.

5 Results and Discussion

A number of examples of the simulation of the single-core multi-cluster model have been examined to establish its performance under various workload conditions. The first case was performed for an 8-single-core cluster system with message length (M) = 32 flits, flit length (F) = 256 bytes and 512 bytes. The second case was performed with the same 8-single-core cluster system and the same flit length (F) = 256 bytes and 512 bytes but with longer message length (M) = 64 flits. The X axis denotes the traffic rate, while the Y axis indicates the communication latency.

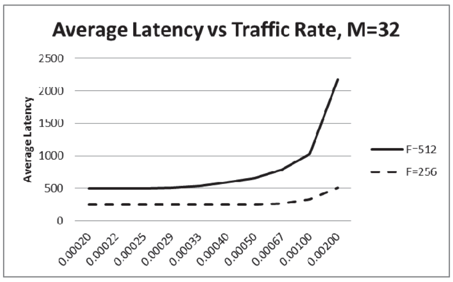

Fig. 6. Average latency of 8-cluster system with M=32 flits, F=256 bytes and 512 bytes.

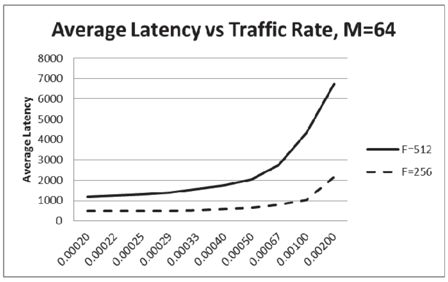

Fig. 7. Average latency of 8-cluster system with M=64 flits, F=256 bytes and 512 bytes.

Simulation experiments have revealed that the results obtained from the single-core multi-cluster architecture closely match the results from the model of multi-cluster architecture presented by Javadi et al. [22], when compared. The results have shown that as the traffic rate increases, the average communication latency increases following the assumptions that the messages are delayed by having to wait for resources before traversing into a network. At low traffic rates, latency will approach zero-load latency. The results confirm that the simulation model is a good basis to measure the communication latency for a large-scale cluster, and can be extended to multi-core multi-cluster architecture.

6 Conclusion and Future Work

This paper has presented architecture for measuring the performance of communication networks in Multi-Core Multi-Cluster Architecture (MCMCA). Preliminary stage of the research involved the development of the single-core multi-cluster simulation model. Simulation experiments have been conducted to evaluate the single-core multi-cluster and baseline results were produced. Simulation results demonstrated that the simulation model is a good basis to measure the communication latency for a large-scale cluster, and can be extended to MCMCA.

Our future work will be developing a simulation model for MCMCA. Experiments will be run in simulation to investigate the model's performance under various configurations. This will provide communication network performance results for comparisons to be made between the model of the MCMCA and models of existing cluster architectures [15, 22]. The simulation will measure communication latency of a cluster when applying multi-core processor technology, under a multi-cluster architecture environment. The approach taken and accuracy of the simulation outcome will make it a good reference for predicting the performance behaviour of MCMCA.

References

1. Intel. (1997, Moore's Law and Intel Innovation. Available: http://www.intel.com/about/companyinfo/museum/exhibits/moore.htm?wapkw=moore+laws
2. D. Geer, "For Programmers, Multicore Chips Mean Multiple Challenges," Computer, vol. 40, pp. 17-19, 2007.
3. T. W. Burger, "Intel Multi-Core Processors: Quick Reference Guide," 2005.
4. Admin. (2009, TOP500 Highlights - November 2009. Available: http://www.top500.org/lists/2009/11/highlights
5. M. Soryani, M. Analoui, and G. Zarrinchian, "Improving inter-node communications in multi-core clusters using a contention-free process mapping algorithm," The Journal of Supercomputing, pp. 1-26, 2013/04/10 2013.
6. E. W. Bethel and M. Howison, "Multi-core and many-core shared-memory parallel raycasting volume rendering optimization and tuning," International Journal of High Performance Computing Applications, vol. 26, pp. 399-412, November 1, 2012 2012.
7. W. J. Dally and B. P. Towles, Principles and Practices of Interconnection Network: Morgan Kaufmann, 2004.
8. G. Shainer, P. Lui, M. Hilgeman, J. Layton, C. Stevens, W. Stemple, et al., "Maximizing Application Performance in a Multi-core, NUMA-Aware Compute Cluster by Multi-level Tuning," in Supercomputing. vol. 7905, J. Kunkel, T. Ludwig, and H. Meuer, Eds., ed: Springer Berlin Heidelberg, 2013, pp. 226-238.
9. S. Ichikawa and S. Takagi, "Estimating the Optimal Configuration of a Multi-Core Cluster: A Preliminary Study," in Complex, Intelligent and Software Intensive Systems, 2009. CISIS '09. International Conference on, 2009, pp. 1245-1251.

10. C. Lei, A. Hartono, and D. K. Panda, "Designing High Performance and Scalable MPI Intra-node Communication Support for Clusters," in Cluster Computing, 2006 IEEE International Conference on, 2006, pp. 1-10.
11. A. Ranadive, M. Kesavan, A. Gavrilovska, and K. Schwan, "Performance implications of virtualizing multicore cluster machines," presented at the Proceedings of the 2nd workshop on System-level virtualization for high performance computing, Glasgow, Scotland, 2008.
12. A. T. Abdelgadir, A.-S. K. Pathan, and M. Ahmed, "On the Performance of MPI-OpenMP on a 12 nodes Multi-core Cluster," in Algorithms and Architectures for Parallel Processing, ed: Springer, 2011, pp. 225-234.
13. H. S. Shahhoseini, M. Naderi, and R. Buyya, "Shared memory multistage clustering structure, an efficient structure for massively parallel processing systems," in High Performance Computing in the Asia-Pacific Region, 2000. Proceedings. The Fourth International Conference/Exhibition on, 2000, pp. 22-27 vol.1.
14. J. H. Abawajy and S. P. Dandamudi, "Parallel job scheduling on multicluster computing system," in Cluster Computing, 2003. Proceedings. 2003 IEEE International Conference on, 2003, pp. 11-18.
15. C. Lei, G. Qi, and D. K. Panda, "Understanding the Impact of Multi-Core Architecture in Cluster Computing: A Case Study with Intel Dual-Core System," in Cluster Computing and the Grid, 2007. CCGRID 2007. Seventh IEEE International Symposium on, 2007, pp. 471-478.
16. L. Chai, "High Performance and Scalable MPI Intra-node Communication Middleware for Multi-core Clusters," PhD, Graduate School of The Ohio State University, The Ohio State University, 2009.
17. A. Varga. (2001). OMNeT++.
18. A. Varga and R. Hornig, "An overview of the OMNeT++ simulation environment," presented at the Proceedings of the 1st international conference on Simulation tools and techniques for communications, networks and systems \& workshops, Marseille, France, 2008.
19. E. Weingartner, H. vom Lehn, and K. Wehrle, "A Performance Comparison of Recent Network Simulators," in Communications, 2009. ICC '09. IEEE International Conference on, 2009, pp. 1-5.
20. S. Mehta, N. Sulatan, and K. S. Kwak, "Network and System Simulation Tools for Next Generation Networks: a Case Study, Modelling, Simulation and Identification," in Modelling, Simulation and Identification, A. M. (Ed.), Ed., ed InTech, 2010.
21. J. Pan, "A Survey of Network Simulation Tools : Current Status and Future Developments," pp. 1-13, 2008.
22. B. Javadi, M. K. Akbari, and J. H. Abawajy, "A performance model for analysis of heterogeneous multi-cluster systems," Parallel Computing, vol. 32, pp. 831-851, 2006.
23. A. Varga. (2011, OMNeT ++ User Manual Version 4.2.2. Available: http://www.omnetpp.org/doc/omnetpp/Manual.pdf