RECONFIGURING DISTRIBUTED APPLICATIONS IN FPGA ACCELERATED CLUSTER WITH WIRELESS NETWORKING

Xin Yu Niu

Dept. of Electrical and Electronic Engineering,
Imperial College London, UK
email: niu.xinyu10@imperial.ac.uk

Kuen Hung Tsoi and Wayne Luk

Dept. of Computing,
Imperial College London, UK
email: {khtsoi, wl}@doc.ic.ac.uk

ABSTRACT

FPGA accelerators are capable of improving computation and energy efficiency of many applications targeting a cluster of machines. In this work, we focus on an FPGA accelerated cluster system modified with a wireless network. Comparing with conventional Ethernet based approaches, the proposed system with wireless networking enables a lightweight and efficient method for the FPGA devices to exchange information directly. Customisable monitoring facilities are developed to support changing a distributed application dynamically at run time. The N-Body simulation application is used to demonstrate the effectiveness and potential of the proposed system. Experiments show that this approach can achieve up to 4.2 times improvement in latency. By applying the proposed inter-FPGA wireless network to the N-Body application, we achieve enhanced power efficiency while fulfilling thermal constraints in all nodes.

1. INTRODUCTION

FPGA (Field Programmable Gate Array) devices are commonly used as accelerators for computational intensive applications. Popular areas for FPGA acceleration include: digital signal processing, physics simulation, logic emulation, cryptographic and finance modelling. To adapt the rapidly growing algorithm complexity and data size, the HPC (High Performance Computing) community has been exploring systems with multiple FPGA devices for a single application. Recent studies also enable the collaboration between heterogeneous accelerators in distributed applications [1].

Besides the computing capability, power and thermal merits are critical for maintaining a stable and scalable production environment. It is desirable to monitor and control these factors for every application such that the physical constraints of the cluster platform are fulfilled. There are various studies considering the power and energy efficiency for individual FPGA accelerators. But a robust and scalable solution for FPGA cluster is not readily available by the best knowledge of the authors.

The aim of this work is to provide an adaptive framework addressing this requirement with the focus on scaling application to cluster platforms. This framework includes means to measure the power and temperature of both FPGA and CPU when the application is running. The collected information is transmitted to a control module promptly. Based on the updated information, a mechanism is used to adjust the configuration of the distributed application.

There are several challenges when realising this framework. First, the time resolution must be high enough for smooth application execution and responsive cluster management. Second, the impact on application performance should be minimised while introducing the monitoring and controlling functions. Also, the solution must be scalable crossing multiple nodes in a cluster and multiple heterogeneous PEs (processing elements) in a node. Finally, the framework should be seamlessly integrated to existing distributed applications.

The major contributions of this work include:

- A modular and customisable wireless network is introduced to allow direct communication between FPGA devices in accelerator clusters. Experiments show that, for the proposed framework, the wireless network is more suitable than the existing Ethernet channel.
- An adaptive framework with facilities for monitoring operation of heterogeneous clusters is developed with the ability to reconfigure the application at runtime for achieving desirable power and thermal conditions.
- The N-Body simulation application is extended to employ these facilities to demonstrate the effectiveness of the approach. Experiments results show quick response and fine grained control using the proposed framework.

2. RELATED WORK

FPGA technology has been used in wireless networks before. An example is an FPGA-based wireless local area network [2], which allows flexible integration of hardware extensions to improve quality of service. The reconfigurability of FPGA technology also enables improvement of power efficiency; it has been shown that, for a Viterbi decoder [3], run-time reconfiguration results in 69% reduction in decoder power consumption over a non-reconfigurable implementation with no loss of decode accuracy.

The power and thermal issues in a cluster system have been investigated. A wireless sensor network has been developed to monitor the power and temperature as part of a cluster management system [4]. For clusters with FPGA and GPU accelerators, a runtime workload scheduling framework [5] has been proposed for Monte Carlo applications.

The research presented in this paper complements existing work described above. Our aim is to show, for the first time, how a modular approach can be adopted to enable a customisable wireless network to be used effectively for improving performance and energy efficiency of a cluster with FPGA-based accelerators.

3. FRAMEWORK OVERVIEW

There are two challenges when designing large-scale computer clusters. First, an appropriate networking technology needs to be found to support communication of not just data, but also control and status information. Second, such networking technology needs to be integrated effectively in the application with minimum overhead.

We propose an innovative framework based on wireless network technology to address these two challenges. The major novelty of our framework is a modular hardware architecture that provides a unified data interface that can support: (a) multiple networking technologies including Gigabit Ethernet through the host CPU and also wireless networking through FPGAs; (b) user kernels which typically consist of circuits that accelerate host applications; (c) customisable monitoring facilities for collecting and analysing status information about performance, temperature and power consumption.

Figure 1 shows a prototype of the proposed framework for a cluster with FPGA based accelerators. In this system, an FPGA accelerator card is installed in each node through the system PCIe interface. While the host software communicates to each other through the Gigabit Ethernet, the FPGA can communicate through an external wireless module. We adopt the 2.4–2.5GHz frequency band, which can support various wireless standards including high-speed IEEE 802.11b and 802.11g, or simpler but slower standards such as ZigBee. The wireless network cooperate complementary with existing Ethernet network to allow heterogeneous channels for direct inter-FPGA communication. The FPGA card is also interfaced to a monitoring module with temperature and power sensors attached to the CPU and FPGA devices.

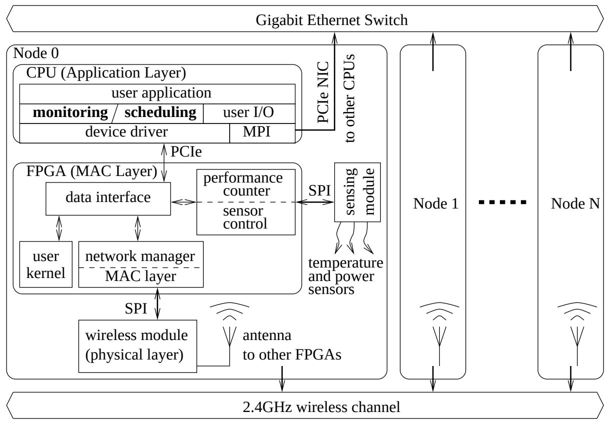

Fig. 1. Cluster structure.

The software and hardware hierarchies are also shown in Figure 1. For the external data flow, the MPI layer is used for transferring application-specific data. The monitoring and scheduling information is sent through the FPGA driven wireless communication channel. For the internal data flow, a unified register-based data interface is developed for the FPGA to enable efficient communication between FPGA modules and high-level application software.

The original user application is extended to include the monitoring and scheduling modules for power and thermal optimisations during runtime. The monitoring and scheduling facilities are included as independent software modules which can be interfaced to user application through C language APIs, and support multiple networking technologies. A soft processor core is used inside the FPGA to support complex control of the wireless and monitoring modules. This modular architecture hides the networking details and monitoring facilities from the user kernels. Hence new user kernels can be developed rapidly for various networks.

The framework can be customised at various levels, such that the constraints and the controlling algorithms can be changed by runtime reconfiguration. At physical layer, various sampling frequency, resolution, modulation and coding algorithms can be configured through the SPI interface. At the medium access control (MAC) layer, packet format, medium access algorithm and acknowledgement strategies can be varied by configuring the unified register file in the MAC module. At the network layer, a network node list is kept. The node authentication, network membership management and timeout detection are implemented here.

4. FPGA DRIVEN WIRELESS COMMUNICATION

In most FPGA accelerated clusters, communication between FPGA go through CPU-driven general purpose network. For packets with small payload, this usually introduces large overhead due to handshaking process, memory copying and networking stacks in the operating systems. Some high performance clusters [6, 7] also install direct connection between FPGA accelerators for higher bandwidth and lower latency. We suggest that an additional wireless communication channel will be beneficial when power and thermal information is to be circulated in the cluster.

In this work, the wireless network is used for transmitting monitored data only. This type of information is usually in form of small packet where application data are usually in large chunks or streaming pipes. For these applications, using FPGA driven wireless network has the following advantages: 1) It has lower and deterministic latency comparing to the software base communication. 2) The management information is separated from the application data for easier integration. 3) It does not rely on any special FPGA hardware feature such as in [6, 7] which both rely on the RocketIO blocks in the Xilinx FPGA and require Infiniband or SATA interface on the accelerator boards. 4) Wireless networks support efficient broadcasting which could benefit protocols or applications involving large amount of broadcasting messages.

A commodity 2.4GHz wireless front-end is used in this study as a rapid prototype. This is sufficient for cluster nodes separate by a few meters in the server racks. In the laboratory environment, the application bandwidth is 1Mbps using GFSK encoding. Due to the strong electromagnetic noise in server room, the Direct Sequence Spread Spectrum (DSSS) technique is used to improve the immunity to noise. The DSSS encoding reduces the practical bandwidth to 125kbps. Information is sent in packet format which has 5 bytes of overhead for preamble, byte count and CRC checksum. To avoid collision when multiple FPGA devices are trying to send data at the same time, a token ring scheme is implemented. In this scheme, an FPGA will send packet only after it acquires the token.

Both the MAC layer control and the network manager are implemented in the FPGA hardware as shown in Figure 1. The packet formatting and token checking are performed in the MAC layer. The network manager provides a control interface and transfers data on top of the MAC layer. These two functions are controlled by a picoBlaze soft core from Xilinx. Only one network manager is activated to support all FPGA accelerators involved in the distributed application. Its main task is to detect transmission timeout and newly available nodes in the network. When a node fails to release the token within a given period, network manager will discard the old token and generate a new one to resume the network operation.

The information on this wireless network can be generated/consumed internally by the FPGA logic or externally by the host software. To access the wireless modules from the CPU side, the software interacts with a set of pre-defined registers mapped to the system memory space. When the FPGA internal logic interacts with the wireless control module, a FIFO is used to avoid data lost.

5. POWER/THERMAL AWARE ADAPTATION

The monitoring modules include integrated performance counters and peripheral circuits to sense the temperature and the supply current of PEs (Processing Elements). As the current information is collected across a small resistor along the 12V power line, a 18V battery supported op-amp is used to amplify the differential inputs. By extracting the idle current, we can get the dynamic power when running a specific application. The performance of PEs is characterised by the reciprocal of processing time for one unit workload in the application.

In response to the cluster status variations, a cluster model should be built to estimate the impacts of current configuration on cluster performance. The power (P), performance (E) and temperature (M) of a PE can be expressed as:

$$P = P_0 + P_1 \times \mathcal{L}$$

$$E = E_0 \times \mathcal{L} \times (1 - \alpha)$$

$$M = M_0 + (\frac{w}{E}) \times \beta + \mathcal{L} \times \gamma$$

Varying parameters in our current model include parallelism ($\mathcal{L}$), workload (w) and overhead of parallel computing (α). The Parallelism is the number of cores or threads running concurrently in a PE. For the power consumption, P_0 is the power consumption in supporting multiple cores within a PE. P_1 is the power consumed by one core. In ideal situation, the performance, E, should scale linearly with the parallelism, $\mathcal{L}$. E_0 is the performance for a single core. The overhead parameter, α, is to model the additional operations such as data distribution, data sharing and synchronisation. Device temperature, M, is proportional to the execution time and consumed power, with M_0 indicating the initial temperature. The β and γ are constant coefficients for computation time (w/E) and parallelism($\mathcal{L}$), respectively. By updating the model during run time, the cluster can estimate the status of every PE. This enables the scheduler to make reasonable decisions on both node level and cluster level. The scheduling mechanism is shown in Figure 2.

The node-level scheduling is executed under three major constraints: maximum device temperature, maximum power consumption and minimum performance. The parallelism of CPU and FPGA will be varied if the measured statuses failed to meet the constraints. Also, the sampled data are used to update the model dynamically in the framework. The reconfiguration parameters are then feed into the model to estimate the temperature, power consumption and performance for next iteration. The PE will then be rescheduled to adapt these changes in the system. After scheduling of the parallelism, the estimated performance will be used by the node-level workload distributor to dynamically balance the execution time of the heterogeneous PEs. As the maximum performance of FPGA accelerators can be 20 times higher than that of CPU, the CPU/FPGA workload ratio is varied iteratively such that the distribution can converge when parallelism of PEs remains stable.

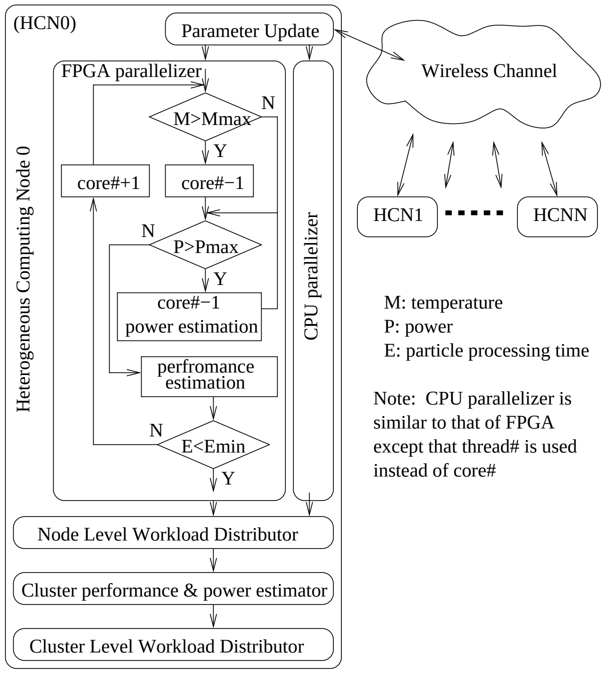
(HCN0)
Parameter Update
Wireless Channel
FPGA parallelizer
M>Mmax
N
Y
core#+1
core#−1
N
P>Pmax
Y
core#−1
power estimation
perfromance
estimation
N
E<Emin
Y
Heterogeneous Computing Node 0
CPU parallelizer
HCN1
HCNN
M: temperature
P: power
E: particle processing time
Note: CPU parallelizer is similar to that of FPGA except that thread# is used instead of core#
Node Level Workload Distributor
Cluster performance & power estimator
Cluster Level Workload Distributor

Fig. 2. Adaptive Distributed Application Framework.

To prevent one single node from becoming the bottleneck of the cluster, the workload for every node is be allocated dynamically. At cluster level, the workload distribution ratio is directly decided by the estimated node performance to provide a quick reaction to node level scheduling.

Since every node has the status of all other CPU and FPGA nodes, the workload redistribution can be done locally without global communication in the cluster. The monitored data will pass through the corresponding communication I/O, such that it can be broadcast in the heterogeneous communication channel.

6. THE N-BODY EXAMPLE

N-Body simulation is commonly used to study interaction between objects with gravitation. The simulation is an iterative process where the 3D position and velocity vectors are updated after combining the total force acting on every individual particle. The computational intensive kernel of the simulation is extracted and implemented in FPGA accelerators in this work. The application is used to demonstrate the efficiency of our proposed adaptive framework.

The computation of acceleration of an individual particle is independent of that of other particles in the same iteration. The accelerating technique is to explore this parallel characteristic of the application. There are three levels of parallelization in our design. Inside each PE, multiple particles are processed concurrently to maximize the silicon utilization rate. This is achieved by employing multiple custom data paths in the FPGA accelerator and multiple concurrent threads in the multi-core host CPU. At the node level, the FPGA and CPU will process different groups of particles concurrently. At the cluster level, the particles are divided and assigned to multiple nodes.

The original design is presented in [1] where the workload is statically distributed for highest performance. This scheme may not be desirable for power and thermal sensitive platforms. In this work, the application is extended to utilise the proposed adaptive framework presented in Section 5. In the extended N-Body design, the power and thermal management is done through controlling the degree of parallelism of the N-Body in various levels. The principle is to reduce the power consumption and heat dissipation by reducing the computation effort in specific PEs.

To facilitate the adaptive framework, signals and control logic are added to the FPGA implementation such that the custom datapath can be enabled or disabled individually during runtime. The dynamic power consumption of the FPGA will be reduced as more parts are disabled and remain static. The modules for wireless communication and status monitoring are also inserted in the FPGA design. The worker threads in the CPU implementation is also modified such that the threads are created dynamically in each iteration.

The monitoring and scheduling functions are inserted in the CPU main loop so that the workload can be redistributed in each iteration. When the N-Body application is launched, the workload is evenly distributed to all participating PEs in the cluster with pre-defined parallelism parameter. The monitoring thread constantly collects and records the power and thermal information when the particles are processed in the CPU and FPGA. Once the local workload is finished, the control thread synchronises and transfers data to other nodes. When all particle information is updated locally, the next iteration can start. Just before starting the next iteration, the workload and parallelism parameters are computed according to the monitored information. In the N-Body application, the number of active CPU threads and FPGA cores are the major reconfiguration parameters. The size of particle group for local processing is also computed automatically based on the expected performance of all PEs.

7. RESULTS

7.1. Environment and FPGA Results

The FPGA accelerator used in this work is the ADM-XRC-5T2 card with a Xilinx Virtex-5 LX330T FPGA. The FPGA design is synthesised using the Xilinx ISE 12.3 tool chain.

The distributed application running on the host CPU is compiled using GCC 4.2.3 and linked against OpenMP and OpenMPI library for intra and inter node parallelization. The cluster is running 64-bit Linux kernel (version 2.6.14). Three of 16 nodes in the cluster are utilised in this study to realize and evaluate the proposed framework.

The FPGA designs in this work are captured using VHDL descriptions. Since the N-Body simulation kernel is tightly coupled with the external memory interface, a single clock is used for both the memory controller and the N-Body kernel. In our 10-core N-Body implementation, the target operating frequency is 200MHz. The number of cores included in the FPGA design is limited by the available DSP blocks used in the floating point operators. Table 1 summarises the implementation results reported by the Xilinx ISE tools. The adaptive version includes the wireless and monitoring drivers with the unified control interface. From the table, we can see that the area overhead for the proposed framework is less than 1%.

7.2. Wireless Performance

Dedicated counters are implemented in FPGA for accurate time measurement in our experiments. The performance of broadcasting the monitored data either through the wireless network or through the MPI over Ethernet is measured. We compare the performance, in terms of the network latency and the stability, when 2 or 3 nodes are involved and when the parallelism of PEs is varied. The packet size is 176 bits, with a 128-bit payload. The measured bandwidth is referred as effective bandwidth.

In the Ethernet-based MPI implementation, the built-in synchronous `MPI_Allgather` model is used. The asynchronous communication model can potentially improve the performance, but constraints on data update will limit the framework granularity. In the wireless implementation, all nodes receive all the broadcast packets, to avoid centralized super-node, or multi-hop communication. This enables each node to have the complete cluster status for reconfiguring the local parts of the distributed application. Also, the cluster performance can scale linearly as the number of nodes in the cluster increases. The maximum and average time for a broadcast operation, as well as the average bandwidth, are recorded during 10 iterations of the N-Body simulation.

As mentioned in previous section, the bandwidth of the wireless front-end is 125kbps when the DSSS coding is employed to increase immunity against the strong noise in server room. Since the payload ratio is 16/21, the maximum achievable bandwidth for the wireless network is 95kbps. As shown in Table 2, where 2N8T means 2 nodes with 8 CPU threads, the performance of the MPI implementation degrades significantly as the CPU is becoming occupied by the workload, while the performance of the FPGA driven wireless network is still near the maximum value even when all 10 cores are occupied. The fluctuation in FPGA wireless bandwidth is due to the additional switch time for from receiver to transmitter. It shows that the average latency for MPI is up to 4.2 times more than that of the proposed wireless network. For the proposed framework, the performance of software-driven wired network is limited by its small packet size, heavy CPU loadings and cluster traffic in the cluster. Running in parallel and separately with the computing cores, the wireless network is much more stable due to its hardware-driven and broadcasting nature. Therefore, It is more suitable for the proposed framework to transmit application data in wired network and circulate cluster status in wireless network.

Table 1. N-Body FPGA implementation results.

resource	non-adaptive	adaptive	difference
LUT	93862(45%)	94818(45%)	956 (0.46%)
FF	115822(55%)	117773(56%)	1951 (0.94%)
DSP	180(93%)	180(93%)	0 (0%)
BRAM	17460(31%)	17556(32%)	96 (0.18%)

Table 2. Network Performance. (N: node, T: thread)

	T_{max} (us)	T_{avg} (us)	bandwidth (Kbit/s)
wireless			
2N1T	2947.84	1472.67	86.92
2N8T	2957.92	1472.06	87.48
3N1T	2947.84	1468.04	87.19
3N8T	2957.92	1472.06	86.95
MPI over Ethernet			
2N1T	1221.46	187.30	1366.80
2N8T	8185.90	1175.26	406.97
3N1T	3373.97	439.24	1784.46
3N8T	11039.13	6246.69	294.66

7.3. Adaptive N-Body Performance

As an example to demonstrate the adaptive framework, status and performance of the nodes are shown in Figure 3 and Figure 4. The power consumption and temperature information are measured by the monitoring module. The temperature constraint is set to $50°C$ for both FPGA and CPU. In our experiment, Node 0 always has lower temperature readings. After applying the adaptive framework, the FPGA temperature is controlled at the $50°C$ level. The workload ratio of FPGA and CPU is changed, in Figure 4, from 17:1 to around 7:1 by the adaptive algorithm as the parallelism of FPGA is reduced to satisfy the temperature constraint.

Figure 5 shows the results of cluster-level scheduling. The performance of PEs is measured as the processing time for one particle. After 20 iterations, the cluster level scheduler dynamically redistributes the workload among the nodes. Results show that this method improves the power efficiency in Node 1 by 21.7% when compared with the non-adaptive implementation. It maintain the power efficiency when work-

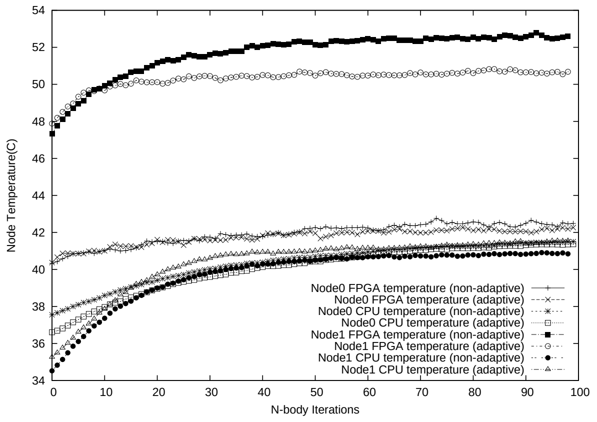

Fig. 3. Node level temperature control.

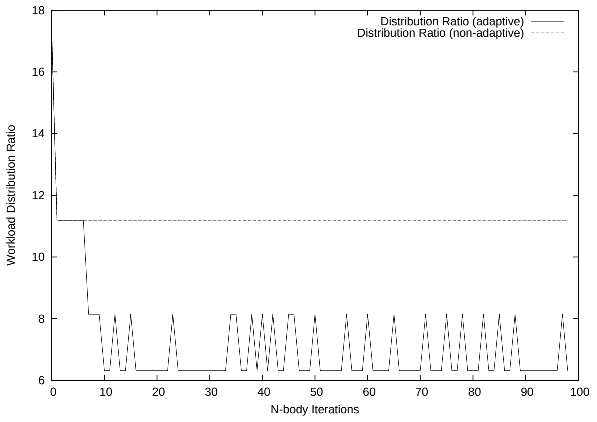

Fig. 4. Node 1 workload scheduling.

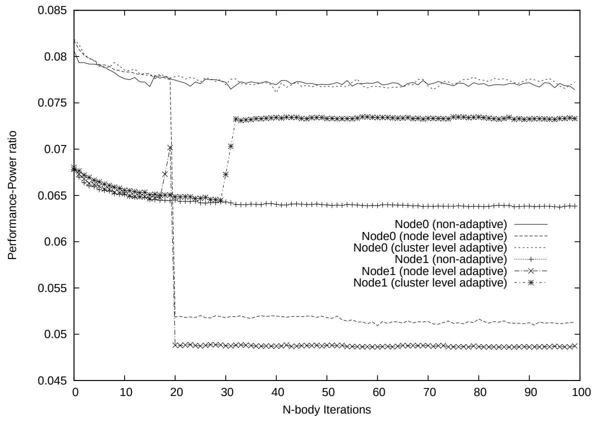

Fig. 5. Adaptive power efficiency.

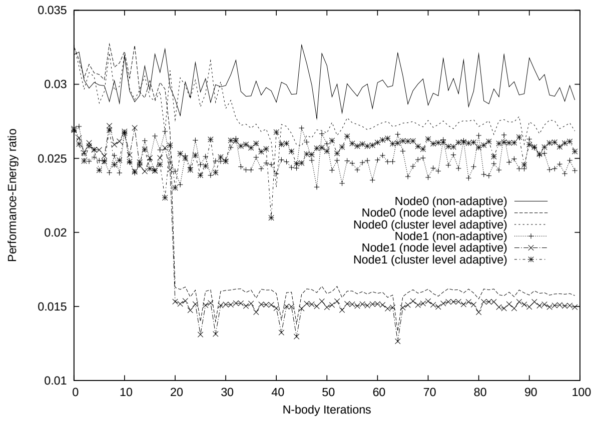

Fig. 6. Adaptive energy efficiency.

load is fixed on node level.

Figure 6 shows that energy efficiency is improved for Node 1, and is slightly reduced for Node 0. While further experiments are needed to confirm the overall trend, our adaptive framework appears to have a negligible impact on cluster energy efficiency while satisfying defined temperature constraints.

8. CONCLUSION

In this paper, we presented a novel framework for power and temperature aware distributed application development targeting FPGA accelerated clusters. This framework is based on a wireless network which has not been used in previous FPGA clusters. Experimental results show that it is more stable and suitable for the proposed adaptive framework. Application developed in this framework can maintain the overall energy efficiency with power and thermal constraints. Current and future research includes performing extensive experiments to explore energy efficiency issues for clusters with accelerators, supporting frequency scaling and partial-reconfiguration in FPGA for improving energy optimization, and extending our approach to cover various wired and wireless network technologies and applications.

Acknowledgement
The support of Imperial College London Research Excellence Award, UK Engineering and Physical Sciences Research Council, Alpha Data and Xilinx is gratefully acknowledged.

9. REFERENCES

[1] K. H. Tsoi and W. Luk, “Axel: A heterogeneous cluster with FPGAs and GPUs,” in *Proc. International Symposium on Field-Programmable Gate Arrays*, 2010, pp. 115–124.

[2] M. Wouters *et al.*, “Real time implementation on FPGA of an OFDM based wireless LAN modem extended with adaptive loading,” in *Proc. Solid-State Circuits Conference*, 2002, pp. 531–534.

[3] R. Tessier *et al.*, “A reconfigurable, power-efficient adaptive viterbi decoder,” *IEEE Trans. on VLSI Systems*, vol. 13, pp. 484–488, 2005.

[4] F. Hu and J. J. Evans, “Power and environment aware control of Beowulf clusters,” *Journal of Cluster Computing*, vol. 12, pp. 299–308, 2009.

[5] A. H. Tse *et al.*, “Efficient reconfigurable design for pricing

asian options," in *Proc. Int. Workshop on Highly Efficient Accelerators and Reconfigurable Technologies*, 2010.

[6] R. Baxter *et al.*, "Maxwell - a 64 FPGA supercomputer," in *Proc. Conference on Adaptive Hardware and Systems (AHS)*, 2007, pp. 287–294.

[7] R. Sass *et al.*, "Reconfigurable computing cluster (RCC) project: Investigating the feasibility of FPGA-based petascale computing," in *Proc. Symposium on Field-Programmable Custom Computing Machines*, 2007, pp. 127–140.