Be Kind, Rewind

Checkpoint & Restore Capability for Improving Reliability of large-scale Semiconductor Design

Igor Ljubuncic
Intel Corporation, Israel
igor.ljubuncic@intel.com

Avikam Rozenfeld
Intel Corporation, Israel
avikam.rozenfeld@intel.com

Andrew Goldis
Intel Corporation, Israel
andrew.goldis@intel.com

Ravi Giri
Intel Corporation, India
ravi.a.giri@intel.com

***Abstract*—Intel's chip design run in a large-scale globally distributed environment with 600,000 cores. In the current semiconductor market scenario, a combination of factors such as time to market pressure, explosive growth in the mobile market segment and upcoming new markets has led to a significant increase in the demand for and reliability of computing resources. Checkpointing is a capability that can make a significant improvement in improving reliability, however, there is no mature solution that allows periodic snapshots of running compute jobs for replay them at a later time in a consistent manner in a large scale environment.**

Intel IT has partnered with the Northeastern University (NEU) Distributed Multi-Threaded Checkpointing (DMTCP) team to improve their checkpoint & restore solution for the design computing environment. This paper elaborates on the innovative technological breakthroughs, industry-academy partnership as well as the open-source contribution.

Keywords—Intel; Information Technology; Engineering Computing; Checkpoint & Restore; Checkpointing; Distributed MultiThreaded Checkpointing; DMTCP; CPU design

I. Introduction

Silicon design industry is one of the large consumers of high performance computing systems driven by the large scale simulations required to address the growth in complexity due to Moore's Law. More recently, the explosive growth in mobile devices which are seeing very complex integration of digital and analog capabilities into an SoC (system on chip) and the expected explosion of volume from the 'Internet of Things' that will bring silicon to everyday objects are driving higher demand for computation. Finally, the competitive market scenario is ensuring that 'Time to Market' pressure on all companies, which drives a need for high levels of availability and reliability of the computing environment.

Intel, being the largest semiconductor company, is generally affected by these trends to a greater extent and earlier than most others in the industry. Intel's state-of-the-art processor design takes place in a large, global setup with >600,000 cores running >40 million jobs every month. Some of these jobs are mission-critical, time-consuming processes that must successfully complete. If they are interrupted for any reason, subsequent reruns must be executed from the start, resulting in a significant delays and waste of project resources. This is particularly true when the results of certain jobs are part of a project's critical path and when other jobs depend on them. These critical tasks then become a bottleneck for the overall execution throughput; failures of any one of these jobs may delay the entire project, and even impact TTM (Time to Market).

In the current technological reality, the execution of tasks in this environment is basically an all-or-nothing approach. Flows that experience irrecoverable errors will crash, and they will have to be executed from the start, wasting precious compute time.

With approximately 600,000 cores, 5 PB of memory, and 24 PB of distributed storage, the Intel IT environment supports roughly a million concurrent regression jobs in a global batch setup across 40 sites daily. The potential for wastage, or conversely, savings is immense.

One approach towards improving reliability and availability is to harden the computing environment by increasing redundancy and investing failover capable systems – most such solutions are proprietary and are for the most part, only able to take care of issues affecting individual compute nodes and not issues that may be affecting the computing environment at large.

Design teams should ideally be able to periodically take snapshots of the jobs, and replay them at a later time, thus improving their ability to debug, branch or replay critical-path executions. Unfortunately, the existing solutions do not possess a robust and generic capability allowing varied applications to stop and resume execution in an opportunistic manner. However, there are several potential technologies that could be developed to allow this mode of work.

Virtualization is one possible solution. It is generic, and usually operating system and application agnostic, however licensing support can be very costly, and there is an additional layer of complexity involved, not to mention performance impact. Furthermore, there are scalability issues since it might be necessary to spawn an instance of a virtual machine for every specific job in order to be suspend and restore them independently. This has also been tested and implemented in specific use cases and scenarios where performance is not as crucial and it has been able to mitigate the risk of single point failure very effectively [1].

Application checkpointing is another alternative, with low to no performance impact but with relatively lower maturity in the industry today. There are two basic approaches that can be used for application checkpointing: in kernel space and in user space. Initially, the Linux community sought to integrate checkpointing into the mainline kernel. However, after the kernel checkpoint/restore was rejected by the KSUMMIT 2010 due to significant complexity involved [2] [3], the development switched to the user space implementation, with Checkpoint/Restore in Userspace (CRIU) [4], as the notable candidate technology. It is currently under development, and available for testing in kernels 3.11 and above. In most data centers and industry environments, the practical implications of this requirement is that very few enterprise-ready operating systems can support CRIU, due to legacy constraints and relatively older kernel versions in use.

Outside the mainline kernel development, third–party software projects have adopted both methods. Notable technologies include Berkeley Lab Checkpoint/Restart (BLCR) [5] and Distributed MultiThreaded CheckPointing (DMTCP) [6], developed by the Northeastern University, MA. The former uses a hybrid kernel/user implementation. A special kernel module is loaded into memory, and it is used to intercept the application system calls. On the other hand, DMTCP does not modify the user's program or the operating system, and resides in the user space only.

A third way to accomplish some checkpoint & restart capability is through custom application-specific solutions [7].However, these are normally not scalable beyond specific applications and require a higher effort to develop and use.

A comparison between the solutions is listed in Table I below:

TABLE I. CHECKPOINTING TECHNOLOGY LANDSCAPE

	Virtualization	**Application checkpointing**	**Application specific**
Platform independence	Yes	Yes[1]	Yes
Licensing support	Yes	No	No
Cost	High	Low	Low
Scalability	No	Yes	Yes
Complexity	High	Medium	Low
Performance	Medium	High	High
Maturity	High	Low	Medium

After a thorough analysis of the present technologies and their relative advantages [8], Intel IT opted to further explore the application checkpointing solutions as a proof of concept for the design community.

Several technologies were examined, including BLCR and DMTCP. An overview of capabilities and comparison between the two technologies in available in Table II (fully implemented feature scores 1, partially implemented with limitation scores 0.5, non-implemented / unavailable features scores 0).

[1] Some kernel version restrictions

TABLE II. COMPARISON BETWEEN DMTCP AND BLCR SOLUTIONS

Feature	DMTCP	BLCR
Workload Agnostic	0.5	1
Support for perl scripts	1	0
Support for interactive sessions (ttys, vnc)	1	0
Support for process groups and sessions	1	0
IPC (pipes) support	1	0
Support for network resource usage	1	0
Support for device files and /proc use	1	0.5
Support for multi-threaded processes	1	1
Support for distributed processes	1	0.5
Non-kernel dependent (User Space)	1	0
Process ID Virtualization	1	0
Checkpoint Image Compression	1	0
Extensibility - ability to write extensions	1	1
Final score	**12.5**	**4**

Eventually, DMTCP was selected as the primary candidate, due to faster and more active development, reduced deployment complexity and a great feature set. BLCR requires a kernel module to be loaded and recompiled following each kernel change [5] – that increases the complexity of deployment and maintaining the tool in a large IT environment.

II. Checkpoint & Restore technology

Checkpointing is essentially the ability to save the state of a workload during execution and restart it, either on the same machine or another machine. A task is executed through a wrapper that maintains persistency of its memory state and periodically saves the contents to disk, allowing the task to be restarted at a later time, with minimal loss of context [5].

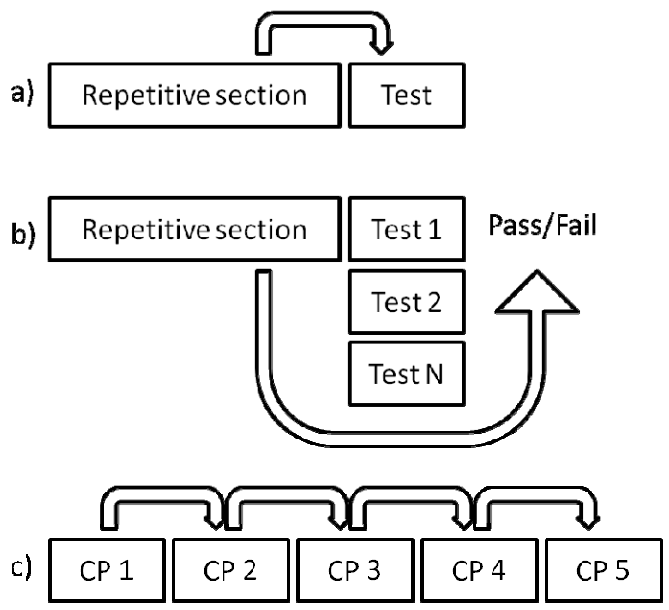
Checkpoint & restore can be used to resolve a range of critical business needs in the design community.

- It can be used for troubleshooting failed simulations / jobs – checkpoints are created periodically, after a job fails the latest checkpoint is used to replay the job and thus improving troubleshooting efficiency and saving computation time.
- It can be used for speculative execution – when multiple simulations/jobs have identical initialization stage, the initialization may be processed once and saved as a checkpoint, while each particular job start

its execution restoring the checkpoint and then executing its unique content.

- Another option is periodic checkpoint & restore used to improve environment reliability, resolving outage scenarios like hardware failures, machine end-of-life, migration of resources, etc.

Fig. 1. Typical use cases of checkpoint & restore technology: a) repetitive execution for troubleshootng b) speculative execution c) environment reliability



III. Partnership & Milestones

Until last year, DMTCP lacked a lot of functionality required to support the huge, complex, demanding environment inside Intel.

We engaged the university team with the clear goal of making DMTCP enterprise ready. The joint work officially kicked off in late 2013, and since we have made some notable progress and great milestones.

The DMTCP partnership currently consists of five major project milestones.

The first one was the establishment of a secure, encrypted Bugzilla system, which allows any Intel engineer in the design space to submit privileged bug reports and enhancement requests to the DMTCP team. In the past six months, the mechanism has been successfully used to escalate a number of key problems in the DMTCP functionality, most of which has been resolved very quickly following the report.

The second milestone was the development of the first stable, mature enterprise-ready release of DMTCP, which can be used by design engineers in the Intel environment. This first version covers a large number of showstopper issues that have so far prevented the business groups from using the checkpoint & restore technology in their flows. Most notably, the following issues have been resolved:

- Restart with different user ID – DMTCP can restore a saved task with the credentials of a different user from the one that checkpointed the task. This is particularly important for debug in groups where many users have access to the same project resources.
- File descriptor referring to a relative path – Another big issue during the restore process was that unless jobs were restarted under the exact same conditions they were saved, process would likely fail to start, because the files needed for the jobs had to exist at the exact location as during the original run. DMTCP Milestone 1.0 resolves this issue by saving relative paths for files and directories used by the job and not only their absolute path, meaning the applications can be moved between directories and mount points, improving portability.
- Mixed 32- and 64-bit applications – Previously, DMTCP could be compiled to support either 32- or 64-bit applications, but not both at the same time. However, many common design workflow involve the use of multiple executables, libraries or tools, where some may be compiled as 64-bit applications, and others may continue to be compiled in the older format of 32-bit applications. Thus, 64-bit applications may call 32-bit applications, and vice versa. DMTCP now correctly checkpoints the full set of processes within a computation, even though it may represent a mixture of 32- and 64-bit processes.
- Performance improvements – DMTCP performance is now well within the required performance specification of 10% above baseline, where baseline is the native performance without DMTCP. The performance of most applications is now indistinguishable when run with or without DMTCP, which is critical for use and acceptance by the design teams. Due to the joint work between Intel and NEU, the previous issue of performance overhead was tracked down to Intel applications that were highly malloc-intensive (frequent memory allocations). The solution employed was to move the wrapper function into an optional DMTCP memory allocation plugin. The plugin is loaded by default, so that the default continues to always ensure deadlock-free operation. The malloc-intensive Intel applications can now disable this plugin to achieve essentially zero overhead.
- Numerous bugs were discovered and submitted, leading to improvements in the DMTCP software.
- We also filed several enhancement requests. For example, it is possible to configure the number of last checkpoints to be saved, whereas till now we had to reconfigure the application in the compilation stage, to be able to save only the last checkpoint or all of them, the former being insufficient and the second taking precious disk space.

Future milestones include DMTCP internal versions 2.0 and 3.0, which will be released later in 2014. There is a strong demand for the ability to checkpoint graphical EDA tools as well as improved network awareness. A large number of design tools have strong network dependencies, including remote filesystem access, communication with other servers and the lease of software licenses. All of these have to be resolved in order to make DMTCP fully enterprise-ready for the strict design needs.

Indeed, the upcoming milestone will support GUI applications and licenses. Milestone 3.0 will improve this support. At the same time, Intel IT will work on developing internal DMTCP expertise to be able to independently support the tool use with the chip design teams. Last but not the least, all of joint work will be publicly released as open-source, further demonstrating Intel's contribution and positive impact on the external community.

However, the most important step in the project is the integration of the DMTCP technology as a part of Netbatch, the EC distributed computing manager software, which controls the Intel IT batch environment. It is massively utilized by Intel's design groups, and there is a great benefit in embedding DMTCP as part of this ecosystem. With checkpointing capabilities in place, it will be possible to allow for far more efficient and manageable resources utilization, and create stable and failure-proof batch-based design tools and execution flows.

IV. Case study – Checkpoint & Restore for Intel's Architecture and Strategic Planning Team

Intel Architecture and Strategic Planning Team is responsible for leading the architectural definition of Intel's next generation products, performing performance analysis of systems and their components. The performance analysis is done during various stages of a product development cycle, using Intel-proprietary software tools that simulate systems behavior while processing billions of instructions.

In order to satisfy the high demand for compute power in the execution of processor design, Intel's IT Engineering Computing division provides an internal batch-like platform distributed computing managing service to design teams. The batch environment is the main horsepower driving the 80K+ core capacity at different geo-locations, allowing effective scheduling and processing of workloads, resources and associative services management. The batch interfaces, which include CLI, API, and GUI elements, allow engineers to implement complex execution flows that may involve different types of tools and ensure a smooth execution in the global distributed environment.

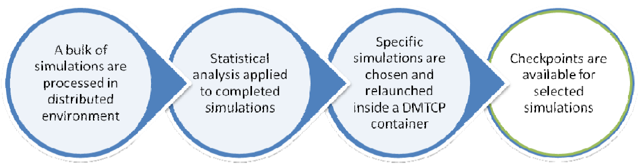
Intel Architecture and Strategic Planning Team uses the batch environment for distributive processing of its simulations, running 400k+ weekly jobs. The processing time for jobs varies and may last up to 18 hours, depending on the number of instructions being submitted for processing. It is often required to reproduce specific job conditions in order to perform a deeper analysis of performance benchmarks, or to examine a failed simulation. In order to ensure that reproduced simulation is completely identical to a failed one, architecture engineers need to rerun the whole simulation, while tracing execution states and applying debugging and analysis tools – which may extend the runtime of a simulation even further.

Implementing a checkpoint & restore solution in the described situation provides obvious benefits both in terms of saving the compute power required to rerun a simulation and, additionally, boosting the productivity and troubleshooting capability of architecture engineers.

DMTCP was successfully embedded into the execution flow of Intel Architecture Team simulations. Due to a small performance penalty incurred when running inside DMTCP, not all simulations are automatically wrapped inside the DMTCP container. As soon a bulk of simulations ends, statistical analysis tools are applied on the results, eliminating simulations that are likely to have common root cause for an issue and selectively picking the simulations that are expected to be examined by an engineer.

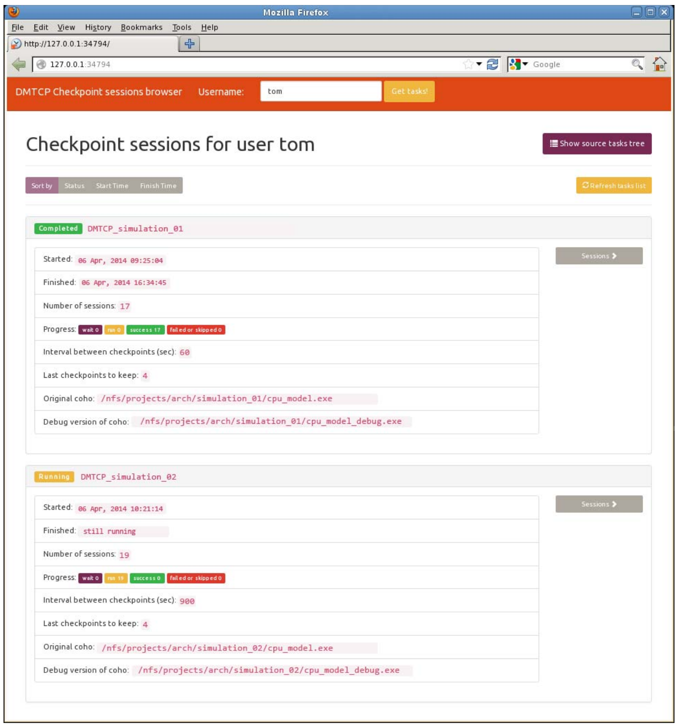
The chosen simulations are then resubmitted to the batch environment, which executes them within a DMTCP container in the distributed environment. As soon as the DMTCP-enabled simulation finishes, there are restore points available for running the simulation closest to the point of interest.

Fig. 2. General flow of using DMTCP checkpoint & restore technology in the execution of the Intel Architecture Team simulations


A bulk of simulations are processed in distributed environment
Statistical analysis applied to completed simulations
Specific simulations are chosen and relaunched inside a DMTCP container
Checkpoints are available for selected simulations

In order to allow for a transparent execution and manageability of the flow, all the information related to the process is stored as part of batch entity, including simulation-related details, DMTCP settings (interval between checkpoints, number of restore points to keep, etc). Here we demonstrate the management interface used by the design engineers.

Fig. 3. Screenshot of DMTCP Checkpoint Sessions Manager implemented for Intel's Arch team.



Checkpoint sessions for user tom

Completed DMTCP_simulation_01

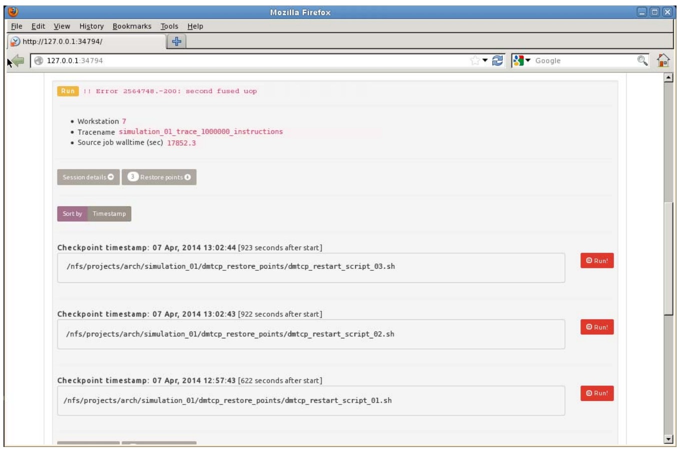
Started: 06 Apr, 2014 09:25:04
Finished: 06 Apr, 2014 16:34:45
Number of sessions: 17
Progress: wait 0 run 0 success 17 failed or skipped 0
Interval between checkpoints (sec): 60
Last checkpoints to keep: 4
Original coho: /nfs/projects/arch/simulation_01/cpu_model.exe
Debug version of coho: /nfs/projects/arch/simulation_01/cpu_model_debug.exe

Running DMTCP_simulation_02

Started: 06 Apr, 2014 10:21:14
Finished: still running
Number of sessions: 19
Progress: wait 0 run 19 success 0 failed or skipped 0
Interval between checkpoints (sec): 900
Last checkpoints to keep: 4
Original coho: /nfs/projects/arch/simulation_02/cpu_model.exe
Debug version of coho: /nfs/projects/arch/simulation_02/cpu_model_debug.exe

After completion, a user can launch the CPU simulator from any of the created restore points.

Fig. 4. Restore points selection and running – screenshot of DMTCP Checkpoint Sessions Manager.



Run !! Error 2564748.-200: second fused uop

- Workstation 7
- Tracename simulation_01_trace_1000000_instructions
- Source job walltime (sec) 17852.3

Checkpoint timestamp: 07 Apr, 2014 13:02:44 [923 seconds after start]
/nfs/projects/arch/simulation_01/dmtcp_restore_points/dmtcp_restart_script_03.sh

Checkpoint timestamp: 07 Apr, 2014 13:02:43 [922 seconds after start]
/nfs/projects/arch/simulation_01/dmtcp_restore_points/dmtcp_restart_script_02.sh

Checkpoint timestamp: 07 Apr, 2014 12:57:43 [622 seconds after start]
/nfs/projects/arch/simulation_01/dmtcp_restore_points/dmtcp_restart_script_01.sh

Until recently, the architecture engineers had to rerun entire flows interactively, with a high risk of missing the failure point and starting the run all over again. With automated DMTCP capabilities, the debugging process is being executed in the batch environment - architecture engineers may start debugging a simulated CPU model from the most recent restore point in effective and productive manner, saving significant amounts of time and computing capacity.

V. Results

Our achievements can probably be broken down into two main buckets. One on hand, we managed to kickstart a cooperation project with the academy and bridge the constraints and bureaucracy gaps while under tight budget. On the other, we have gained some impressive technological results along the way.

As mentioned before, DMTCP is being piloted by Intel's Architecture and Strategic Planning team to greatly improve the debugging efficiency of the CPU simulator – the major tool used for performance simulations of the CPU architecture. With periodic checkpoints every 15 minutes and a robust restore capability, the design engineers are able to skip rerunning the successful part of their 18-hour simulation and launch a restore point that occurs close to a failure. This approach eliminates the need for resources required for the rerun, and improves the debugging efficiency by allowing a repeatable replay of code in the restored sessions.

Furthermore, the cooperation with the university has led to significant improvements in the functionality of DMTCP, including support for mixed 32/64-bit executables and libraries, ability to restart (restore) with a different UID than the one that originally ran the program and created a checkpoint, and ability to use file descriptors referring to a relative path. Most importantly, the co-Intel-developed DMTCP milestone version has a significance reduction in the performance penalty, and it is in line with the Intel IT requirements. Our baseline is less than 10% degradation compared to standard flows, whereas before our involvement, many applications suffered from 20-30% degradation. Table III summarizes the runtime results for a standard debug run and those with DMTCP checkpoints.

TABLE III. WITH DMTCP CHECKPOINTING EVERY 15 MINUTES, THE REPLAY OF A DEBUG RUN IS RESTRICTED TO JUST THE LAST CHECKPOINT RATHER THAN THE WHOLE 18-HOUR RUN

	Without DMTCP	**With DMTCP**
CPU flow simulation runtime (h)	> 18	< 0.25

VI. The future

Within Intel IT, we will focus on the development and enhancement of the DMTCP technology for use with graphical EDA tools, with strong network dependencies. Moreover, we will work on integrating the software into Netbatch, which will then expose the checkpoint & restore capabilities across the entire global batch setup.

There is also additional engagement with third-party vendors to include native DMTCP support in their tools, as well as engagement with super-computing development teams on enabling DMTCP for the Xeon Phi [9] family of products.

VII. Conclusion

Checkpoint & Restore technology has significant merits and business ROI in large compute environments like the Intel IT's global setup. We see a strong demand for this kind of capability with the design teams. The use case with Intel's

Architecture and Strategic Planning team is an important milestone in the overall enablement of the DMTCP software within Intel. Most importantly, our project has led to a number of significant improvements in the core functionality of the technology, which can only be fully realized when coupled with a complex ecosystem like Intel's chip design environment, due to its unique scope and size. Exciting future milestones will include checkpointing for EDA tools, and a complete integration with Intel's batch-like distributed management solutions.

Our partnership with the DMTCP team clearly demonstrates academy-industry cooperation. We hope it will serve as a precedent for future endeavors of similar nature. From the technical perspective, enabling a complete flexible and robust checkpoint & restore solution for the development teams is a revolutionary achievement, creating a new reality for the fast-paced SoC transition happening across Intel. From the strategic point of view, we have exercised significant industry leadership and impact, and managed to work around the tight restrictions that persist in the company today, showing that innovation coupled with real customer needs still define the core of our business.

Acknowledgment

We would like to thank Professor Gene Cooperman and his team at Northeastern University for the close cooperation and partnership in this project.

References

[1] V. Kamath, R. Giri and R. Muralidhar, "Experiences with a Private Enterprise Cloud: Providing Fault Tolerance and High Availability for Interactive EDA Application," in *2013 IEEE Sixth International Conference on Cloud Computing*, Santa Clara, 2013.

[2] J. Corbet, "Linux info from the source," 2 November 2010. [Online]. Available: http://lwn.net/Articles/412749/.

[3] J. Corbet, "Linux info from the source," 19 July 2011. [Online]. Available: http://lwn.net/Articles/452184/.

[4] Parallels; OpenVZ, "Checkpoint/Restore In Userspace," Open Source Software Community, [Online]. Available: http://criu.org/.

[5] P. H. Hargrove and J. C. Duell, "Berkeley Lab Checkpoint/Restart (BLCR) for Linux Clusters," *Journal of Physics* , vol. 46, no. Conference Series, pp. 494-499, 2006.

[6] J. Ansel, K. Arya and G. Cooperman, "DMTCP: Transparent Checkpointing for Cluster Computations and the Desktop," Rome, 2009.

[7] C. V. Walters J, "Application-level checkpointing techniques for parallel programs.," in *3rd ICDCIT pp 221–234*, 2006.

[8] K. Byoung-Jip, "Comparison of the Existing Checkpoint Systems," Watson/IBM, 2005.

[9] Intel Corporation, "Intel® Xeon Phi™ Product Family," Intel Corporation, 2014. [Online]. Available: http://www.intel.com/content/www/us/en/processors/xeon/xeon-phi-detail.html.