Checkpointing and Adaptive Scheduling in HPC: A Case Study on the Aznavour Cluster

Mikayel Gyurjyan
IIAP NAS RA
Yerevan, Armenia
e-mail: mikayelg@gmail.com

***Abstract*— This paper presents a practical and extensible approach to implementing checkpointing on the Aznavour HPC cluster in Armenia using the DMTCP (Distributed MultiThreaded Checkpointing) system, integrated seamlessly with the SLURM workload manager. We describe the benefits and methodology of transparent checkpointing and propose a design for future extensions, based on models for adaptive job scheduling that respect task priorities, resource demands, and waiting time constraints.**

***Keywords*—High-performance computing, checkpointing, adaptive scheduling, SLURM, DMTCP, Aznavour cluster, queueing models.**

I. Introduction

High-performance computing (HPC) clusters face increasing demands for fault tolerance and efficient resource utilization. Checkpointing — the ability to save and restore program states — is critical for long-running computational jobs, especially in systems like the Aznavour cluster, Armenia's national supercomputing infrastructure [10].

We focus on integrating checkpointing mechanisms into the Aznavour cluster using DMTCP and SLURM and provide a future roadmap for incorporating advanced queue scheduling strategies based on recent theoretical models.

II. Overview of Checkpointing Systems

Checkpointing systems are essential components of resilient HPC infrastructures, enabling jobs to resume execution from their last saved state in the event of a failure. These tools vary in terms of their integration level, support for parallelism, and ease of deployment. A good checkpointing mechanism must balance **transparency**, **performance overhead**, and **support for complex workloads** such as MPI-based applications.

Several prominent checkpointing systems exist:

- **BLCR (Berkeley Lab Checkpoint/Restart) [4]**: A kernel-level tool that supports MPI, now considered legacy.
- **DMTCP (Distributed MultiThreaded Checkpointing) [3]**: A user-space library with strong support for MPI and SLURM, without kernel dependencies.
- **CRIU (Checkpoint/Restore In Userspace) [5]**: Widely used for containerized applications in environments like Docker or LXC.
- **OpenMPI-integrated checkpointing [6]**: Previously supported via BLCR but deprecated in newer MPI versions.

Among these, **DMTCP** emerges as the most practical choice for **Aznavour**, given its:

- Compatibility with OpenMPI/MPICH
- Non-invasive integration via job scripts
- Periodic and manual checkpoint support
- Suitability for batch-scheduled environments like SLURM

In subsequent sections, we describe how DMTCP can be integrated into Aznavour's SLURM-based pipeline to deliver robust fault-tolerance capabilities.

III. Implementation on Aznavour Cluster

The **Aznavour cluster**, Armenia's national HPC platform, supports scientific research in physics, biology, climate modeling, and AI. The cluster comprises dozens of compute nodes connected via high-speed interconnects and managed by the **SLURM workload manager**.

In 2025, **DMTCP-based checkpointing** was deployed across all SLURM queues, offering fault-tolerance and flexibility to long-running jobs. The cluster’s architecture is now equipped to support:

- Transparent state saving and job recovery
- Periodic and on-demand checkpointing
- SLURM-native job requeuing and resume functionality

Benefits Observed Post-Deployment

- **Job resilience**: Long simulations (>24h) are now protected from hardware failure
- **Minimal overhead**: Checkpointing introduces minimal overhead, with performance primarily influenced by the checkpoint interval
- **Zero-code integration**: Most MPI applications required no modifications

https://doi.org/10.51408/csit2025_15

- **Simplified job recovery**: SLURM scripts automate restart from last checkpoint
- **Future extensibility**: Hooks available to integrate adaptive scheduling policies

IV. SLURM and Checkpointing Integration

SLURM offers native support for job requeuing (`--requeue`). Integration with DMTCP is achieved by wrapping MPI jobs with `dmtcp_launch`, using `dmtcp_command` for checkpointing, and `dmtcp_restart` to resume jobs. Below is an example script:

```
#!/bin/bash
#SBATCH --job-name=myjob
#SBATCH --output=res.out
#SBATCH --time=01:00:00
#SBATCH --requeue
module load dmtcp
srun dmtcp_launch ./my_mpi_app
```

V. Enhanced Queueing Extensions

We explore advanced queue models, including MFIFO, WTR, and virtual waiting time analytics. They allow SLURM to prioritize tasks based on patience, resource intensity, and fair share, potentially using SLURM plugin extensions or job_submit.lua hooks.

As an example, consider a queueing model with a waiting time restriction (WTR), where jobs are allowed to wait in the queue only for a limited time before being dropped or rescheduled. In SLURM, this can be represented by associating a maximum virtual waiting time with each job and prioritizing resubmissions or reservations accordingly. Mathematically, such systems have been modeled using Markov chains and steady-state probabilities, as shown in [2,7,8]. These models provide a foundation for designing SLURM plugins that dynamically adapt scheduling based on system load and job patience profiles.

VI. Performance Evaluation

Preliminary tests on the Aznavour cluster indicate that checkpointing introduces minimal overhead, with performance primarily influenced by the checkpoint interval, number of processors, memory usage, and task size. These parameters will be explored further to derive scalable performance models and visualizations. No application code changes were needed, and DMTCP handled multi-process MPI jobs gracefully. Reliability of jobs improved significantly, especially for long-running simulations.

VII. Future Work

Next steps include cluster-wide integration with SLURM plugins, GPU job support, predictive scheduling based on waiting time thresholds, and adaptive scheduling via reinforcement learning.

VIII. Conclusion

DMTCP checkpointing on the Aznavour cluster demonstrates the potential of lightweight, transparent recovery tools. Integration with SLURM was straightforward, and future improvements may include queue-aware scheduling using theoretical models from Armenian HPC research.

References

[1] V. Sahakyan, Y. Shoukourian, H. Astsatryan, “About Some Queueing Models for Computational Grid Systems,” *Math. Problems of Computer Science*, vol. 46, pp. 55–58, 2016.

[2] V. Sahakyan, A. Vardanyan, “About Virtual Waiting Time in a Multiprocessor System,” *CSIT Conference*, pp. 111-113, 2023. https://doi.org/10.51408/csit2023_23

[3] DMTCP GitHub repository. [Online]. Available: https://github.com/dmtcp/dmtcp

[4] P. Hargrove, J. Duell, “BLCR for Linux Clusters,” LBNL. [Online]. Available: https://crd.lbl.gov/

[5] CRIU Project Documentation. [Online]. Available: https://criu.org/

[6] “OpenMPI FAQ - Fault Tolerance”. [Online]. Available: https://www.open-mpi.org/faq/?category=ft

[7] V. Sahakyan, A. Vardanyan, “The Steady State Distribution for M|M|m|n Model with the Waiting Time Restriction,” *Math. Problems of Computer Science*, vol. 54, pp. 34–40, 2020.

[8] M.Gyurjyan, V. Sahakyan, “Queues modelling of the multimachine computing system”. *Mathematical Problems of Computer Science*, Transactions of IIAP NAS RA, vol. 23, pp. 166-174, 2004.

[9] H. Astsatryan, T. Grigoryan, M. Gyurjyan, V. Sahakyan, Yu. Shoukourian, “Development of Web Environment for Efficient Exploitation of Linux Cluster” Computing Resources. *Proceedings of the Seventh International meeting on VECPAR’06*, July 10-13, 2006, Rio de Janeiro, Brazil.

[10] Armenian National Supercomputing Center. [Online]. Available: https://anscc.sci.am