Scalable Fine-Grained Gang Scheduling for HPC Systems with Unreliable Broadcast Synchronization Mechanisms

Hiroki Ohtsuji[1], Erika Hayashi[1], Reika Kinoshita[1], Masahiro Miwa[1], Eiji Yoshida[1]

1) Fujitsu Ltd.

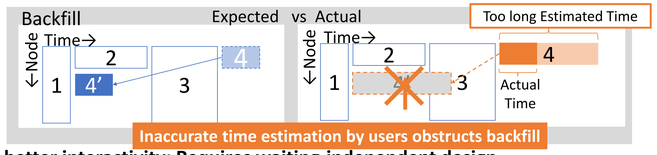
Abstract

The demand for interactivity on HPC systems is increasing, primarily driven by new HPC users from the AI/ML research area. Traditional HPC users are accustomed to waiting for job execution on a batch scheduling system while new users prefer an interactive terminal such as Jupyter Notebook. To address these evolving requirements, enhancing interactivity is essential. Fine-grained gang scheduling is one potential solution for this problem. This poster presents **a scalable inter-node synchronization mechanism** that facilitates well-time-aligned synchronization message delivery through broadcast communication for fine-grained gang scheduling in HPC systems. The mechanism improved the application performance by 2.7 times compared to the existing implementation, when concurrently running two parallel applications on 128 computing nodes using a 500ms time slice duration.

Background

Increased demand of interactivity of HPC system

- AI/ML application scaling is driving new user adoption on HPC systems.
- HPC systems using batch queues: Even for short jobs, there can be long wait time.
  - Backfill can reduce wait time, but it requires users to accurately estimate job execution time



For better interactivity: Requires waiting-independent design

- Resources can be divided in **space** or **time**

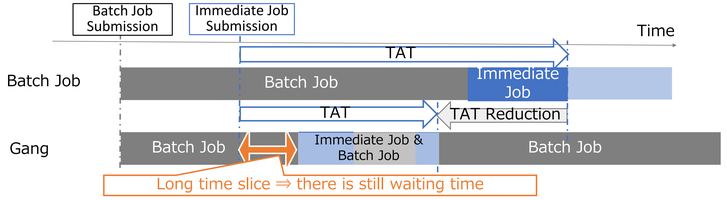
- Space sharing

- <issue> MPI and other tools do not support rescaling during execution.
  - Inability to perform large-scale execution limits the benefits of HPC.

- Time sharing

- Switching in short time slices: Reducing the turnaround time (TAT) for short processing times.
- <issue> Significant performance degradation in multi-node execution.

Time sharing fits the requirement of interactivity, particularly scalable **gang scheduling** (inter-node synchronized time sharing)



Shortening time slice is essential for better interactivity in large-scale HPC environments

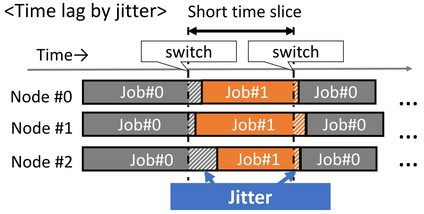
Challenges in Current Implementations

Why is inter-node synchronization crucial for scalable gang scheduling?

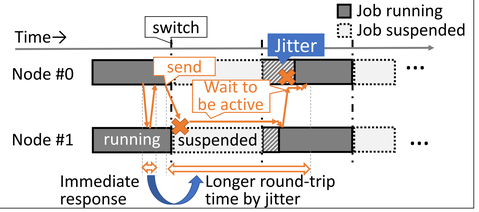
- **Time lag between jobs is caused by control communication jitter** (Left figure)
  - As time slices become shorter, the proportion of time lag becomes larger.
  - As the number of nodes increases, the jitter is increased due to the difficulty of simultaneously sending messages to all nodes using **Unicast communication.**
- **Time lag between jobs degrades the application performance** [1] (Right figure)
  - Jitter significantly affects communication performance of parallel applications
  - If data is sent to a suspended node due to jitter, the sender node will not receive a reply until the next job switch occurs.
    ⇒ The sender node wastes execution time while waiting for a response.

Improving inter-node synchronization accuracy is necessary

<Time lag by jitter>



<Impact of time lag on application performance>



Solution and Implementation

Utilizing broadcast communication for inter-node synchronization

<Current> Control communication in job schedulers using **unicast**
<Solution> Employing **broadcast communication** for synchronization

Solution:

- The controller broadcasts synchronization messages.
- The agent program running on computing nodes receives messages.
  - Sends SIGSTOP/SIGCONT signals to PIDs corresponding to the target Job ID.

Implementation:

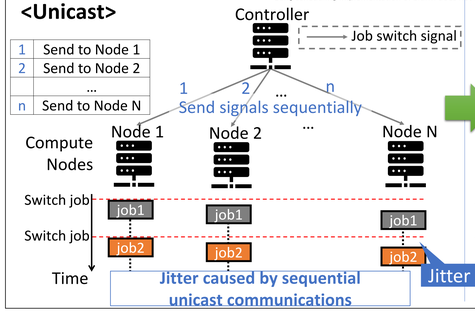
- Integrated with Slurm 22.05 *[3]. Works on partitions with oversubscription.

* Gang scheduling implementation of Slurm does not support the time slice shorter than 5 seconds.

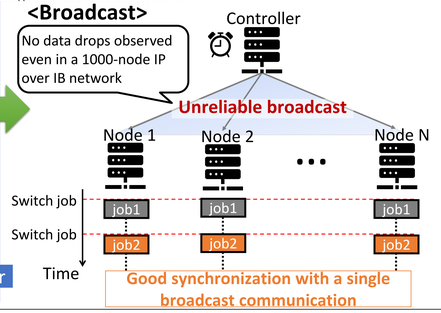
Good synchronicity of Unreliable Broadcast
+
Data drop in InfiniBand (IB) network is rare [2] and results in an absence of only one job switch on the node, which **doesn't cause errors**.

Benefits of good synchronization outweigh drawbacks of unreliable communication

<Unicast>



<Broadcast>



Evaluation

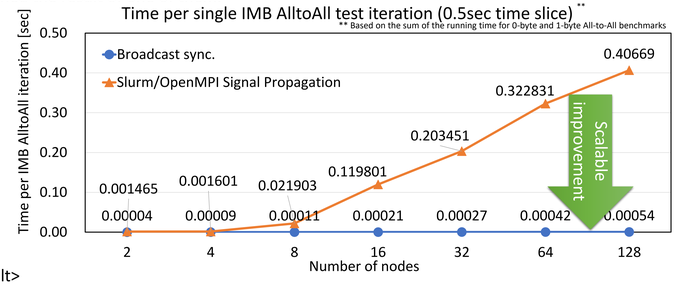
Evaluation of communication performance:

- All-to-All communication performance with Intel MPI benchmarks
  - To comprehend the significance of accuracy in synchronizing inter-node job switching.
  - Comparison: **Unreliable broadcast sync.** and Slurm's signal propagation mechanism with OpenMPI.
  - Disabling the gang scheduling feature of Slurm (controlling processes from outside the scheduler).
  - Running two IMB benchmark programs simultaneously and obtained the result of one of them.

Evaluation Environment
Compute Node:
FUJITSU Supercomputer PRIMEHPC FX700
CPU: FUJITSU A64FX Arm8.2-A SVE
48cores @ 2.0GHz
RAM: 32 GiB HBM2
Network interface: InfiniBand HDR100
Interconnect: Full-bisection bandwidth fat-tree
OS: Rocky Linux 8.5
Slurm: 22.05
OpenMPI: 4.1.5

Time per single IMB AlltoAll test iteration (0.5sec time slice) **

** Based on the sum of the running time for 0-byte and 1-byte All-to-All benchmarks



<Result>
Performance of communication is significantly influenced by the accuracy in node-level job switching.
- The performance gap ratio increases as the number of computing nodes.

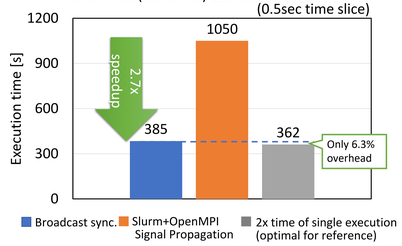
The unreliable broadcast synchronization mechanism is scalable

Application performance benchmark

- Running two GROMACS (benchRIB)[4, 5] jobs on the same resources (128 nodes).
- Switching jobs with three different methods. (Comparison: **Broadcast sync.**, Slurm's signal propagation mechanism, and doubled time of single run)

<Result>
- The proposal mechanism considerably shortens the execution time compared to Slurm/OpenMPI signal propagation thanks to high-fidelity synchronization.
- Broadcast sync introduces only a slight overhead compared to doubled time of single run.

GROMACS (benchRIB) 128nodes execution time (0.5sec time slice)



Broadcast synchronization enables scalable interactivity on large-scale clusters

References

[1] A. Gupta *et al.*, 1991. The Impact of Operating System Scheduling Policies and Synchronization Methods of Performance of Parallel Applications. SIGMETRICS Perform. Eval. Rev. 19, 1 (Apr 1991), 120-132. **[2]** A. Kalia *et al.*, 2016. FaSST: Fast, Scalable and Simple Distributed Transactions with Two-Sided (RDMA) Datagram RPCs, USENIX OSDI 16, 185-201. **[3]** Slurm Workload Manager, (online), available from https://slurm.schedmd.com **[4]** M. J. Abraham *et al.*, 2015. GROMACS: High performance molecular simulations through multilevel parallelism from laptops to supercomputers, SoftwareX, Vol. 1-2, pp. 19-25. **[5]** Max Planck Institute for Multidisciplinary Sciences, G ottingen : A free GROMACS benchmark set, (online), available from https://www.mpinat.mpg.de/grubmueller/bench