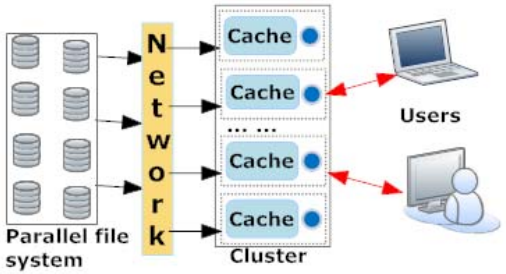
ScalScheduling: Scalable Scheduling for MPI-based Data Analytic Programs

Jiangling Yin, Andrew Foran, Xuhong Zhang and Jun Wang (jyin@eecs.ucf.edu)
Department of Electrical Engineering & Computer Science, University of Central Florida

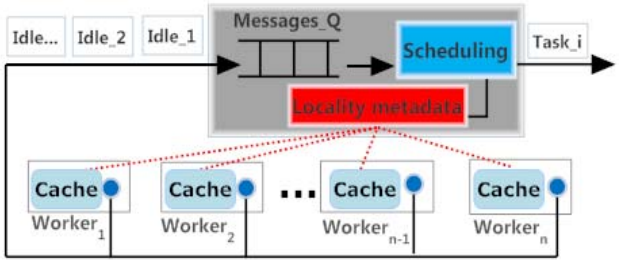
Background

- For **interactive** data processing, a job should be finished in seconds
  - > Gene sequence search (mpiBLAST)
  - > Interactive visualization (ParaView)
  - > Data analysis (Log processing)
- To mitigate **data movement** overhead: a local disk cache implemented at compute node, enables running process to **reuse locally stored history data**.

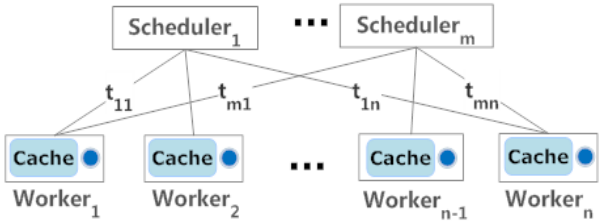


Problems

- To reuse historical data, a **scheduler** with data locality consideration is a must.
- However, scheduling a task causes hundreds of milliseconds latency when taking **data locality** into consideration.



ScalScheduling Architecture

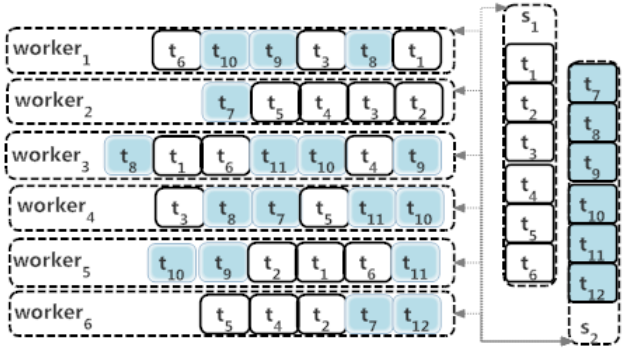


- **Multiple schedulers**: keep track of the data processing tasks.
- Each **worker** process: a novel Modulo-based priority method to schedule its **own local tasks** (as long as local data exists).
- If no local data exits, a scheduler will assign a remote task to an idle worker.

Modulo-based Method

$$x = \lceil \frac{z}{n} \rceil, y = (z-1)\%n + 1, b = \lceil \frac{f}{n} \rceil$$

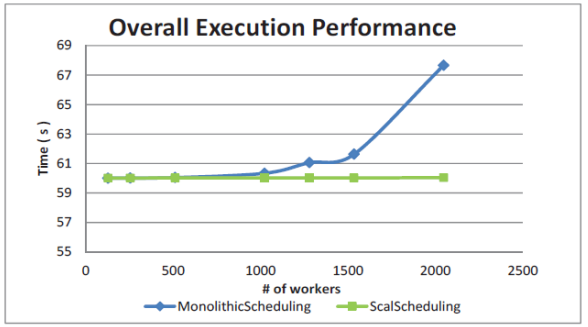
$$prio(z) = b \times ((y+n-i)\%n) + (x+b-i\%m)\%b$$



- A example with f=12, m=2, n=6 and the local tasks sorted with the Modulo-based method.

Experimental Results

- Program **execution time comparison** on Marmot (NSF PRObE cluster)



Conclusion

- A **scalable scheduling** architecture to support task request/assignment for **a large number of worker processes** running in parallel data intensive applications.
- **Performance improvement** over monolithic scheduling architectures
- We will incorporate ScalScheduling into real workloads.

Acknowledgement

This material is based upon work supported by the National Science Foundation under the following NSF program: Parallel Reconfigurable Observational Environment for Data Intensive Super-Computing and High Performance Computing (PRObE).