Pacific Northwest NATIONAL LABORATORY

Proudly Operated by Battelle Since 1965

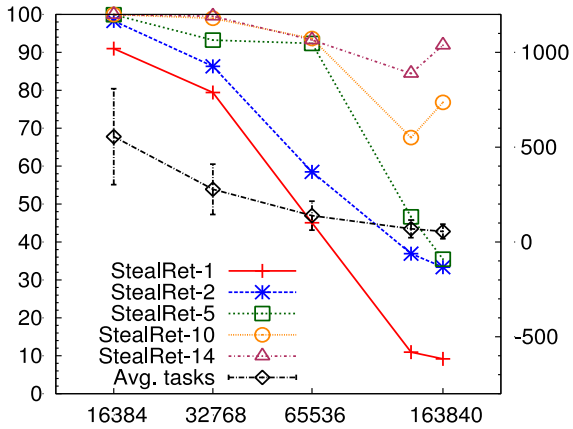
TASCEL

TASCEL (pronounced “tassel”) is a framework to study the design of algorithms to address the challenges associated with programming abstractions supporting finer-grained concurrency. It uses an active message framework built on MPI. MPI allows quick prototyping and evaluation on various platforms with minimal porting effort. The active message framework enables the design of supporting algorithms that are concurrent with ongoing execution (e.g., load balancing or fault recovery concurrent with application execution). TASCEL supports various threading modes, progress semantics, together with SPMD and non-SPMD execution, so as to be representative of a variety of useful execution environments.

Following is some recent work on algorithms for finer-grained concurrency abstractions using the TASCEL library.

SCHEDULING AND LOAD BALANCING

J. Lifflander, S. Krishnamoorthy, and L. Kale. "Work stealing and persistence-based load balancers for iterative overdecomposed applications." HPDC'12.

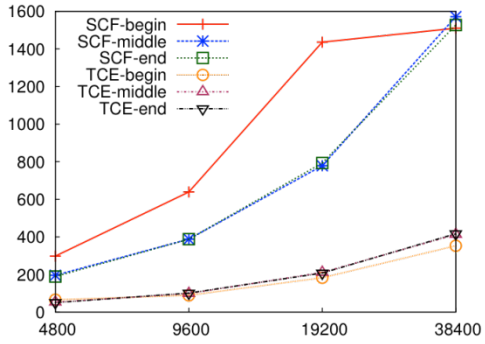


Parallel efficiency (left axis) & average tasks per worker (right) vs core count using retentive stealing on Intrepid (TCE benchmark)

Applications often involve iterative execution of identical or slowly evolving calculations. Such applications require incremental rebalancing to improve load balance across iterations. We evaluated two distinct approaches to addressing this challenge: persistence-based load balancing and work stealing. We developed a hierarchical persistence-based rebalancing algorithm that performs localized incremental rebalancing. We also developed a retentive work stealing algorithm optimized for iterative applications on distributed memory machines. Retentive work stealing exploits persistence to incrementally rebalance work and reduce the overhead associated with random stealing.

FAULT TOLERANCE

W. Ma and S. Krishnamoorthy. "Data-driven fault tolerance for work stealing computations." ICS'12.



Factor of reduction in re-executed tasks as compared to collective rollback with respect to core count on Hopper

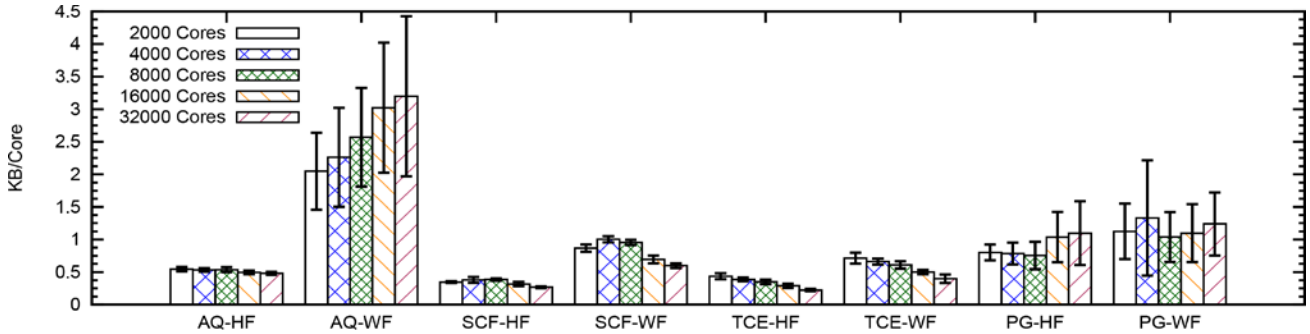
We have designed fault tolerance mechanisms for task parallel computations operating on global data. We developed three recovery schemes that present distinct trade-offs: lazy recovery with potentially increased re-execution cost, immediate collective recovery with associated synchronization overheads, and non-collective recovery enabled by additional communication. We employ distributed-memory work stealing to dynamically rebalance the tasks onto the live processes. We demonstrated that the overheads (space and time) of the fault tolerance mechanism were low, the costs incurred due to failures were small, and the overheads decreased with per-process work at scale.

U.S. DEPARTMENT OF ENERGY

TRACING WORK STEALING

J. Lifflander, S. Krishnamoorthy, and L. Kale. “Steal Tree: low-overhead tracing of work stealing schedulers.” PLDI'13

Work stealing is inherently flexible and can tolerate variations due to faults, power, of system noise anticipated on exascale systems. However, such a flexible approach to dealing with unbalanced distribution of work results in seemingly irregular computation structures, complicating the study of the runtime behavior of work stealing schedulers. Typical approaches to studying work stealing often resorted to tracking information on each individual task. Given that the number of tasks can be orders of magnitude greater than the number of processor cores; this approach can quickly become intractable. We have developed an approach to efficiently trace async-finish parallel programs scheduled using work stealing with low time and space overheads. We demonstrated the broader applicability of this work, in addition to replay-based performance analysis, through two use cases: the optimization of correctness tools that detect data races in async-finish programs; the design of retentive work stealing algorithms for recursive parallel programs.



Space overhead in tracing help-first and work-first schedulers on OLCF Titan (AllQueens, SCF, and TCE benchmarks)

CONTINUING WORK

We continue to investigate algorithms for load balancing and resilience. We are investigating algorithms to effectively support various threading models, memory management strategies, and exploitation of the machine hierarchy. Finer-grained programming abstractions other than work stealing are also being evaluated.

ADDITIONAL PAPERS RELATED TO TASCEL

- J. Lifflander, P. Miller, and L. V. Kale. "Adoption protocols for fanout-optimal fault-tolerant termination detection." PPoPP'13.
- J. Daily, S. Krishnamoorthy, and A. Kalyanaraman. "Towards scalable optimal sequence homology detection." ParGraph'12.
- A. Panyala, D. Chavarria, and S. Krishnamoorthy. “On the use of term rewriting for performance optimization of legacy HPC applications.” ICPP’12.

COLLABORATORS AND CONTRIBUTORS

Pacific Northwest National Lab:
Sriram Krishnamoorthy, Daniel Chavarria, Jeff Daily, Wenjing Ma

Ohio State University:
P. Sadayappan

University of Illinois, Urbana-Champaign:
Jonathan Lifflander, Laxmikant Kale

Washington State University:
Ananth Kalyanaraman

CONTACT: sriram@pnnl.gov

MORE INFO:
http://hpc.pnl.gov/projects/tascel

ABOUT PNNL

Interdisciplinary teams at Pacific Northwest National Laboratory address many of America's most pressing issues in energy, the environment and national security through advances in basic and applied science. PNNL employs 4,500 staff, has an annual budget of nearly $1 billion, and has been managed for the U.S. Department of Energy by Ohio-based Battelle since the laboratory's inception in 1965. For more information, visit the PNNL News Center, or follow PNNL on Facebook, LinkedIn and Twitter.

Pacific Northwest
NATIONAL LABORATORY

Proudly Operated by Battelle Since 1965