The Relentless Execution Model for Task-uncoordinated Parallel Computation in Distributed Memory Environments

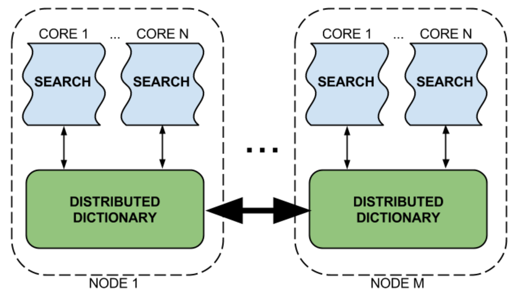
Lucas A. Wilson

Department of Computer Science
The University of Texas at San Antonio
San Antonio, Texas, U.S.A.

Motivation and Problem Statement

Modern High Performance Computing (HPC) systems consist of many thousands of individual servers. As these systems become larger and more complex it will become increasingly difficult, if not impossible, for sufficient numbers of nodes to remain available during parallel computations. **Existing execution and programming models are not well suited to future distributed memory parallel systems where hardware reliability cannot be guaranteed.**

The Relentless Execution Model (REM) is a dataflow-inspired execution model which relies on uncoordinated parallel processes that interact with a distributed, eventually-consistent key-value store (*dictionary*), where the values are the data operated on, and the keys inform the runtime task scheduler which operations should be performed on that data.



Contributions

- A theoretical model for the execution of parallel algorithms in a task-uncoordinated fashion, which is capable of executing tightly-coupled, data-dependent algorithms in distributed memory environments where hardware volatility may result in occasional-to-frequent loss of computational processes.
- A domain-specific language, StenSAL, which allows for rapid, easy-to-understand encoding of explicit stencil algorithms targeting the proposed task-uncoordinated model.
- A series of offline optimizations, which can be performed mechanically by the StenSAL compiler, which enable multi-threaded worksharing and vectorization without explicit input from the programmer.
- A mathematical means of determining the minimum amount of memory (and therefore the minimum number of nodes) required to be available in order for a particular stencil algorithm to maintain computational progress.

References

Lucas A. Wilson and Jeffery von Ronne, "A Task-uncoordinated Distributed Dataflow Model for Scalable High Performance Parallel Program Execution," *Parallel Computing: Systems & Applications*, Elsevier. DOI:10.1016/j.parco.2015.10.013

Lucas A. Wilson and Jeffery von Ronne, "StenSAL: A Single Assignment Language for Relentlessly Executing Stencil Algorithms," in proceedings of the Workshop on Optimizing Stencil Computations (WOSC), October 2014.

Lucas A. Wilson and Jeffery von Ronne, "A Distributed Dataflow Model for Task-uncoordinated Parallel Program Execution," in proceedings of the Seventh International Workshop on Parallel Programming Models and Systems Software for High End Computing (P2S2), September 2014.

Lucas A. Wilson and John A. Lockman III, "Relentless Computing: Enabling Fault-tolerant, Numerically Intensive Computation in Distributed Environments," in proceedings of The 2011 International Conference on Parallel and Distributed Processing Tools and Applications (PDPTA'11), July 2011.

Components of a Relentless Program

1. a finite set T_p of tasks to be performed;
2. a finite set L_p of labels, which are used as keys in the shared dictionary;
3. a set $R_p \subseteq L_p$ of result labels whose association with values completes the REM program's execution;
4. a surjective function $producer_p : L_p \rightarrow T_p$ that maps dictionary labels to the task that produces the value for that label;
5. a function $requires_p : T_p \rightarrow \mathcal{P}(L_p)$ that maps each task to the labels of the inputs the task requires before it can be executed; and
6. a function $computes_p : T_p \rightarrow (L_p \nrightarrow V) \nrightarrow (L_p \nrightarrow V)$ that maps each task to the partial function it computes.

The Relentless Task Search Algorithm

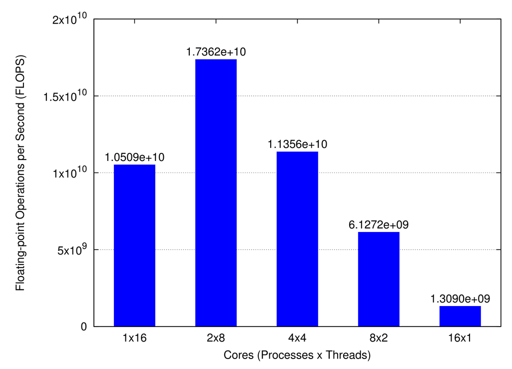
for all $r \in Result$ **do**
    $SOLVE(r)$
end for
function SOLVE(label)
    if $label \notin dom(Dictionary)$ **then**
        $task \leftarrow PRODUCER(label)$
        $Missing \leftarrow \{l \mid l \in REQUIRES(task) \wedge l \notin dom(Dictionary)\}$
        for all $m \in Missing$ **do**
            $SOLVE(m)$
        end for
        $Inputs \leftarrow \{(l \mapsto v) \mid (l \mapsto v) \in Dictionary \wedge l \in REQUIRES(task)\}$
        $Dictionary \leftarrow Dictionary \cup COMPUTES(task)(Inputs)$
    end if
end function

StenSAL: Explicit Stencil Algorithms in a Single Assignment Language

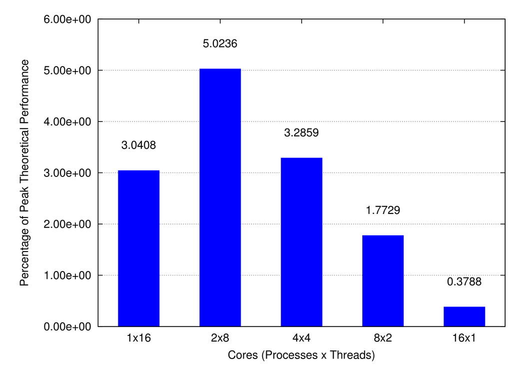
StenSAL allows for fast expression of explicit stencil algorithms in a functional style that is easily translated to REM. The following example is the expression of a task for solving a 3-point central stencil over a 1-dimensional domain:

```
task updateTemp
  creates u(x)(t) as newVal
  for
    x=1:maxX-1
    t=1:maxT
  using
    u(x-1)(t-1) as left
    u(x)(t-1)   as center
    u(x+1)(t-1) as right
  code
    newVal = center + alpha *
      (left + right - 2 * center)
  end code
end task
with
  alpha=0.25
```

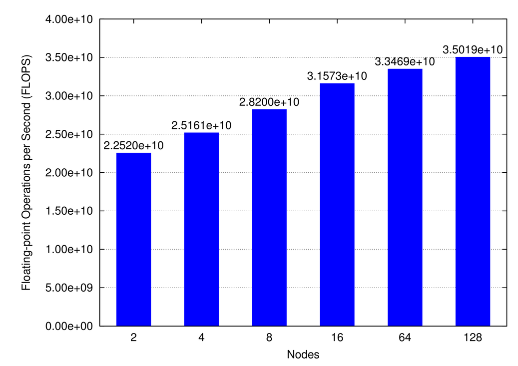
REM Performance



Single-node Throughput (FLOPS)

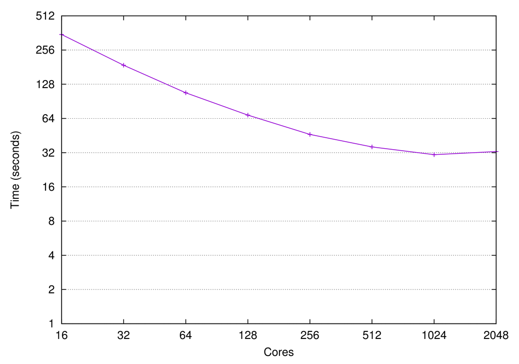


Single-node % of Peak Performance

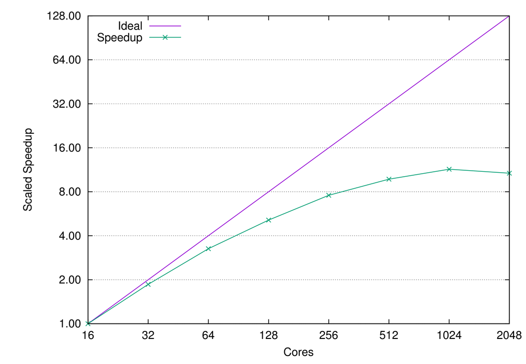


Multi-node Throughput (FLOPS)

Performance Using $120k^2$ Coalesced Element Tasks



Walltime (secs.)

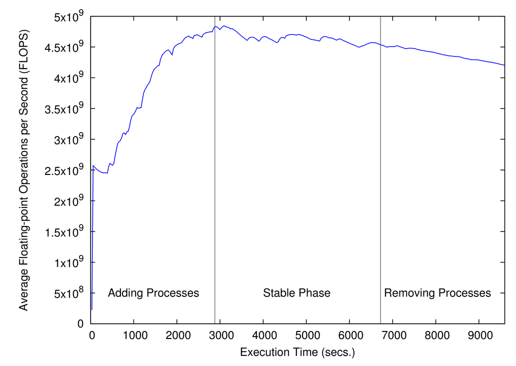


Scaled Speedup

Multi-node Scaling using $16k^2$ Coalesced Element Tasks

Elastic Parallel Computation

The example stencil problem was solved such that processes are first added to a running computation, and then later removed from the same running computation. Average throughput in FLOPS shows that REM can efficiently add and remove computational tasks while continuing execution.



Average throughput of elasticity experiment

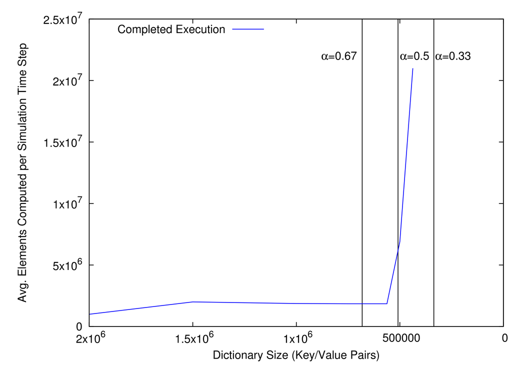
Minimum Memory Required for Computational Progress

The minimum dictionary size for a particular problem can be calculated as:

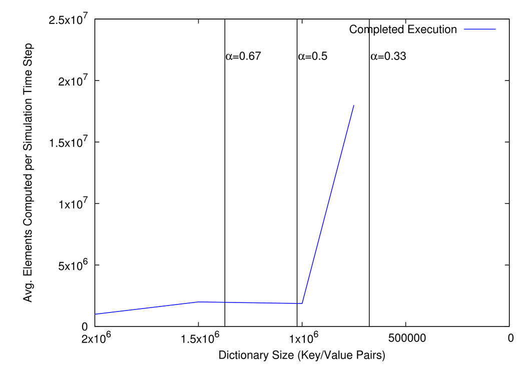
$$\lfloor D \rfloor = \alpha \prod_{i=1}^{|c|} \left(\mid \mathcal{T}_{k+1}^{c_i} \mid + \Gamma \right)$$

$$\Gamma = \max_{\tau \in \mathcal{T}_{k+1}} deg^{-}(\tau)$$

$\mathcal{T}_{k+1}^{c_i}$ is the set of tasks when traversing the i^{th} dimension of coordinate tuple c for sequencing index $k+1$, Γ is the retention factor, defined as the maximum in-degree of any task τ in time step $k+1$, and α is an additional term added for curve fitting.



9-point 2-D stencil



27-point 3-D stencil

Results of dictionary restriction tests