Strong Scalability Analysis and Performance Evaluation of a SAMR CCA-based Reacting Flow Code

SAMR: *Structured Adaptive Mesh Refinement*

Multiscale algorithm for Cartesian meshes

Pros:

Less grid points, less compute time.

Preservation of numerical accuracy.

Cons:

Load Balancing problems.

Lack of scalability.

The Problem: Algorithms ensuring efficient resolution and numerical accuracy may not be scalable. Scaling analyses of SAMR simulations are rare.

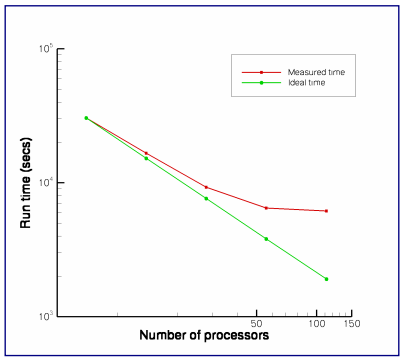
The Objective: Identify the non-scalability sources.

Methodology:

1. Choose a test problem where the initial and final states of the computational domain are vastly different so that:
   i. It poses a challenge to domain partitioning.
   ii. The quality of the partitioning affects scalability.
2. Analyze timing measurements and domain decomposition specifics to identify the causes of lack of scalability.

Problem Statement:

- Global problem size is held constant.
- Virtual machine expanded in steps of two, starting with 7 processors.



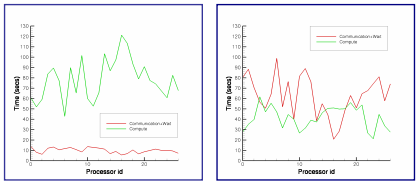
For more than 28 processors experimental results deviate a lot from the linear scale up behavior. What are the possible causes?

- Excessive transfer times caused by network saturation.
- Synchronization costs due to non-optimal domain and load partitioning.

Results-Analysis:

We pick two processor topologies, 28 and 112, and analyze.

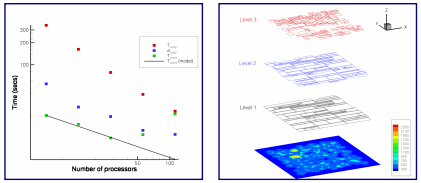
We have a dynamically adaptive algorithm. Are all intermediate states equally scalable or non scalable? No they are not.



Timestep 40 Timestep 80

Communication and Compute time distributions for the 28 processor run.

Scalable State *(Timestep 40)*

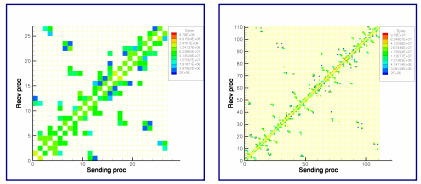


On the left: Average compute time (T_{comp}) and standard deviation (σ_{comp}), average communication time (T_{comm}) for 7…112 processors at timestep 40. On the right: Snapshot of the mesh hierarchy at timestep 40.

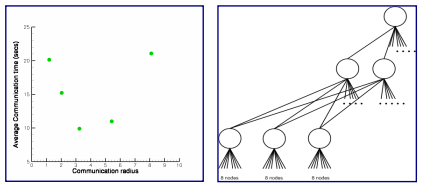
T_{comp} dominates over T_{comm} (which includes data transfer and synchronization times).
We model communication time (transfer rate bound model) as: $T_{comm}(model) \sim t_{tr}$, where t_{tr} is the transfer data time:

$$t_{tr} \propto \frac{1}{\sqrt{p}}, i.e. \frac{(T_{comm})_{p_2}}{(T_{comm})_{p_1}} = \sqrt{\frac{p_1}{p_2}}$$

Linear scalability suffers for p 56. Why?



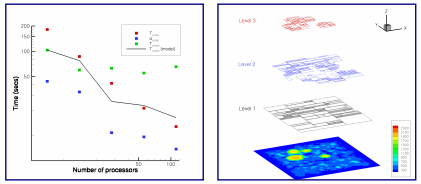
Communication patterns for p=28 (left) and p=112 (right) at timestep 40.



On the left: Average communication time versus average communication radius for 5 different runs (min p=7, max p=112) at timestep 40. On the right: Clos network schematic.

The average communication radius is 3.2 for p=28 and 8.1 for p=112. By comparing their communication patterns we see that as the number of processors increases they communicate further in the virtual machine.
As the communication pattern radius approaches 8 nodes and substantial communication occurs over tier-1 switches the communication times increase.
Note: $T_{comp} >> T_{comm}$, so Timestep 40 is scalable.

Unscalable State *(Timestep 80)*

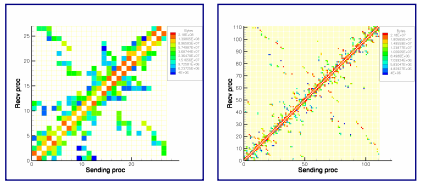


On the left: Average compute time (T_{comp}) and standard deviation (σ_{comp}), average communication time (T_{comm}) for 7…112 processors at timestep 80. On the right: Snapshot of the mesh hierarchy at timestep 80.

T_{comm} dominates over T_{comp} (T_{comm} includes data transfer and synchronization times).
Since the mesh has become visibly sparser the increase in T_{comm} is NOT due to data transfer times.
We model communication time (synchronization bound model) as:
$T_{comm}(model) \sim t_{sync}$, where t_{sync} is the synchronization time:

$$t_{sync} \propto \sigma_{comp}, i.e. \frac{(T_{comm})_{p_2}}{(T_{comm})_{p_1}} = \frac{(\sigma_{comp})_{p_2}}{(\sigma_{comp})_{p_1}}$$

The synchronization bound model only holds for p=7 and p=14.
For $p > 28$, $\sigma_{comp}/T_{comp} > 0.25$ and the non-linear effects of σ_{comp} dominate. This analysis will be part of future work.



Communication patterns for p=28 (left) and p=112 (right) at timestep 80.

Sandia National Laboratories

Sophia Lefantzi, Jaideep Ray and Sameer Shende