

Automation of Determination of Optimal Intra-Compute Node Parallelism

Scalable Tools Workshop

PRESENTED BY:

Antonio Gómez

agomez@tacc.utexas.edu

James C. Browne

Why?

- Many applications using MPI for intra-node parallelism
- Not all loops in the code are the same
- Improve resources utilization, get highest intranode parallelization
- But still, make it as easy as possible for users

Using PerfExpert for this

- PerfExpert
 - Under development since 2008
 - Show users something simple
 - We don't look for best performance, but for good performance
 - Several different tools integrated into PerfExpert
 - Compilation, Measurement, Instrumentation, Analysis, Recommendation
- Continuous improvements
 - Analysis parallelization
 - Load imbalance
 - Vectorization reports
- Support for KNL

<https://github.com/TACC/perfexpert>

What are we trying to do

- Help users characterize their codes
- Create a list of most critical loops and code sections with:
 - Information about LCPI
 - Highest possible degree of parallelism of that loop/section
- Expect changes in the code by the user
- Rerun analysis
- Automate as much as possible
- And this is only intra-node

Find critical sections

- Use LCPI
 - HPCToolkit/VTune under the cover (Measurement)
- LCPI metric is calculated for each code section (Analysis)
- Metrics are modified depending on the processor
- Still adding support to KNL
 - Consider MCDRAM
 - Detect memory mode

LCPI

- LCPI (Local Cycles Per Instruction)
- Several metrics associated to the main one
 - Processor dependent
 - Sandy Bridge
 - Data
 - TLB
 - ...

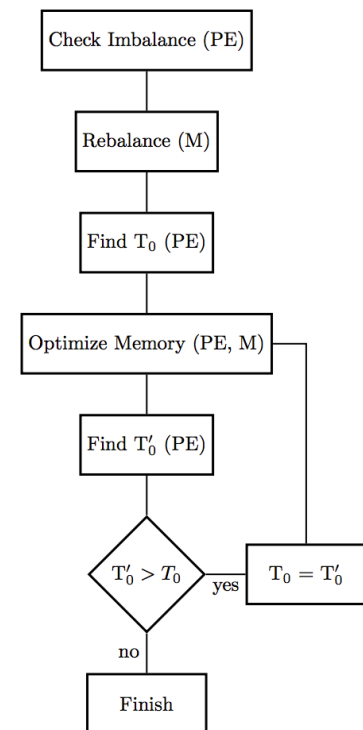
$$\text{LCPI}_{\text{Data}} = \frac{\text{L1_HIT} * \text{L1_lat} + \text{L2_Hit} * \text{L2_lat} + \text{L2_Miss} * \text{Mem_lat}}{\text{TOT_INS}}$$

What's the idea?

- Start with MPI applications
- Find critical loops
- Optimize the code
- Annotate highest degree of parallelism
- When no further optimization, introduce OpenMP
- Reoptimize
- But do this considering the highest degree of parallelism possible (empirical value) and the overhead introduced by OpenMP

Automated workflows

- MPI Workflow
 - Many applications still use MPI for intra-node parallelization
- Idea
 - Find critical sections
 - Identify scalability for those sections
 - Improve memory access pattern
 - Rerun scalability
 - Repeat if necessary



Estimation Workflow

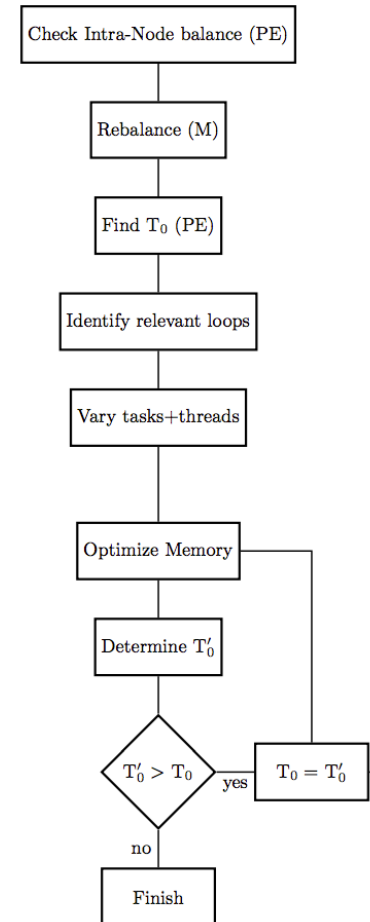
- For the main loops in the code, identify their LCPI
- Get max. theoretical speedup and compare with achieved
- Decide whether to continue or not

LCPI - Sandy Bridge

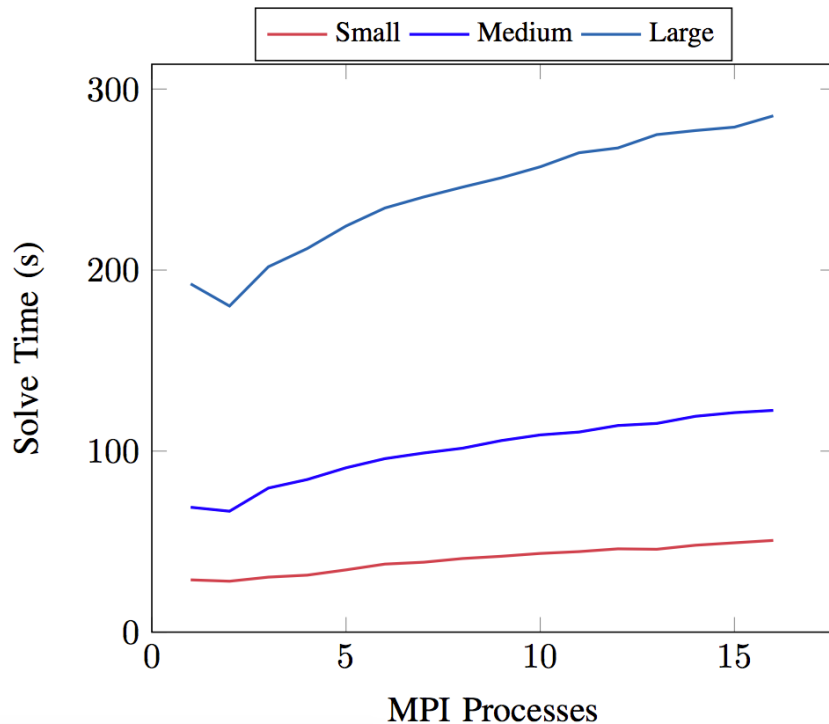
LCPI					Optimal Degree of Local Parallelism
0	$\leq$	x	<	0.5	16
0.5	$\leq$	x	<	0.6	14
0.6	$\leq$	x	<	0.65	12
0.65	$\leq$	x	<	0.7	10
0.7	$\leq$	x	<	0.75	8
0.75	$\leq$	x	<	0.8	6
0.8	$\leq$	x	<	0.9	5
0.9	$\leq$	x	<	1	4
1	$\leq$	x	<	1.5	3
1.5	$\leq$	x	<	2.5	2
2.5	$\leq$	x			1

Hybrid Workflow

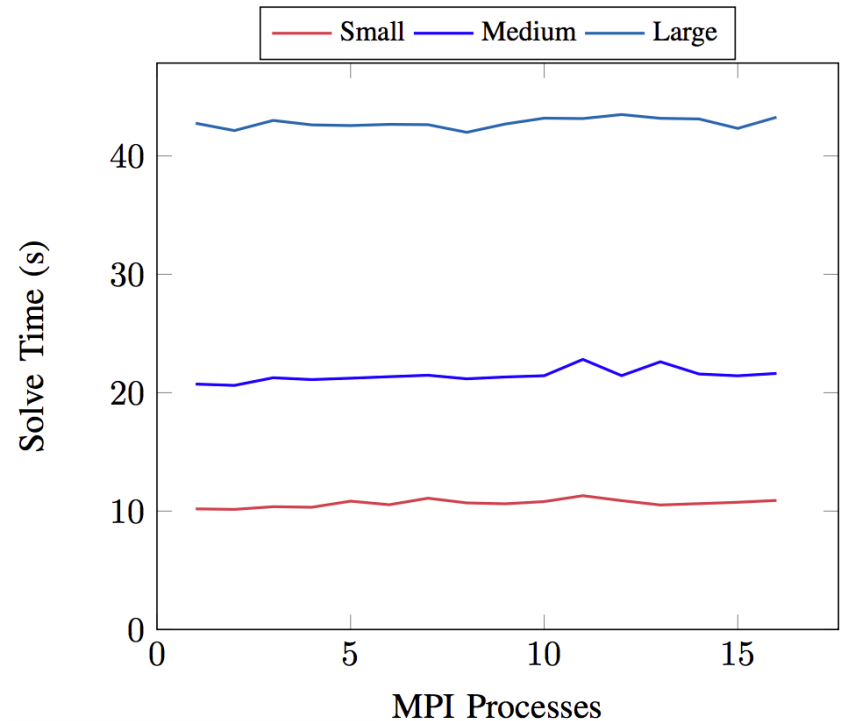
- Consider OpenMP overhead
 - Identify a **threshold** that specifies whether adding OpenMP is beneficial or not
 - Add OpenMP
 - Calculate LCPI
 - Modify memory access pattern
 - Calculate LCPI
 - Check if benefit and compare different with the threshold



Some Results (SPPARKS)



Original Weak Scalability



Optimized Weak Scalability

Future of PerfExpert

- Lustre counters (IO in general)
- Integration of MPI_T (MPI Advisor)
- Considering OMPT
- Software versioning control
- Extending user interface
- Instrumentation
 - Already doing something (MACPO: memory access pattern)
 - What else?
- Keep it simple
- Promotion!

<https://github.com/TACC/perfexpert>

Something different now

REMORA

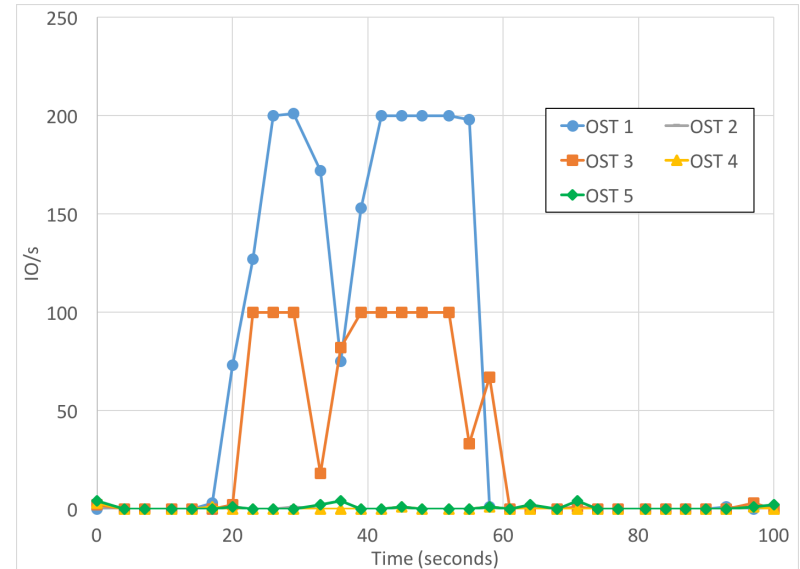
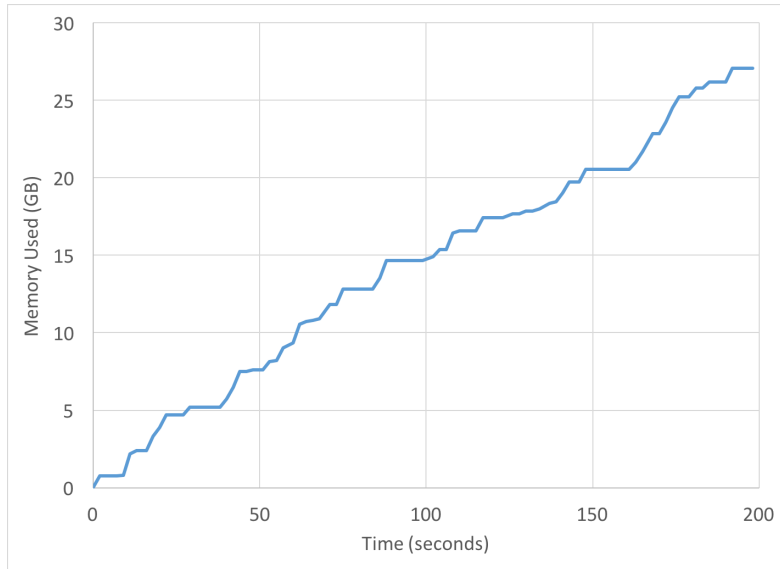
- Monitoring/Profiling tool developed at TACC
- Very simple:
 - Background task on each node
 - Collects:
 - CPU utilization
 - NUMA stats
 - Memory utilization (free, virtual,...)
 - Lustre counters
- Fairly popular tool at TACC systems (XALT)
- Very easy to use, easy to understand

```
$ remora ./myexe
```

```
$ remora mpirun ./myexe
```
- Answers simple questions



REMORA

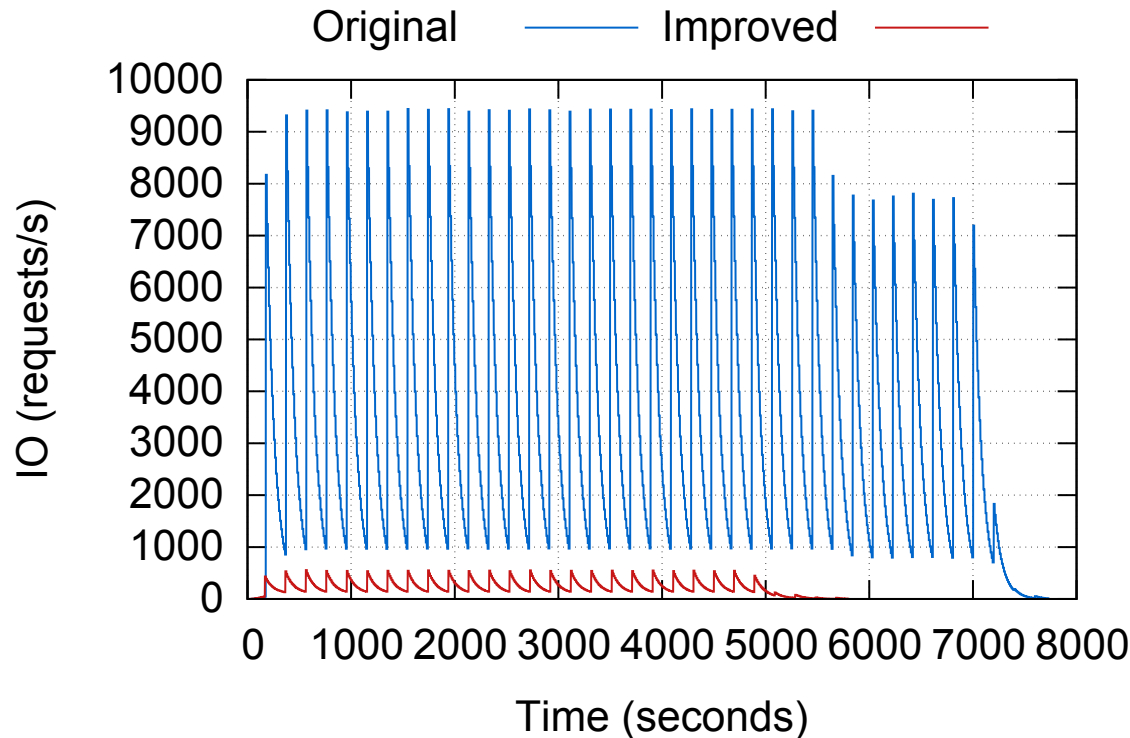


```
===== REMORA SUMMARY =====
Max Memory Used Per Node      : 7.65 GB
Total Elapsed Time            : 0d 0h 1m 9s 176ms

-----
Max IO Load / home1           :      0 IOPS      0 RD(MB/S)      0 WR(MB/S)
Max IO Load / scratch         :      76 IOPS    3011 RD(MB/S)    425 WR(MB/S)
Max IO Load / work            :      0 IOPS      0 RD(MB/S)      0 WR(MB/S)

-----
Sampling Period                : 1 seconds
Complete Report Data           : /lbm_bench/bin/remora_7306879
=====
```

Use Case: More IO



- Original code creating high IO load
- Improved IO: reduce frequency and how it is implemented
- New code: Improved performance. Improved stability of filesystem

Automation of Determination of Optimal Intra-Compute Node Parallelism

Scalable Tools Workshop

PRESENTED BY:

Antonio Gómez

agomez@tacc.utexas.edu

James C. Browne