Hints to improve automatic load balancing with LeWI for hybrid applications

Marta Garcia, Jesus Labarta, Julita Corbalan

Journal of Parallel and Distributed Computing — Volume 74, Issue 9
September 2014

Motivation

- Loss of efficiency
- Hybrid programming models ($MPI + X$)
- Manual tuning of parallel codes (load-balancing, data redistribution)

The X (in this paper)

OpenMP

- Directives to annotate parallel code
- Fork/join model with shared memory
- Number of threads may change *between* parallel regions

SMPSs (SMPSuperscalar)

- Task as basic element
- Annotate *taskifiable* functions and their parameters (in/out/inout)
- Task graph to track dependencies
- Number of threads may change *any time*

DLB and LeWI

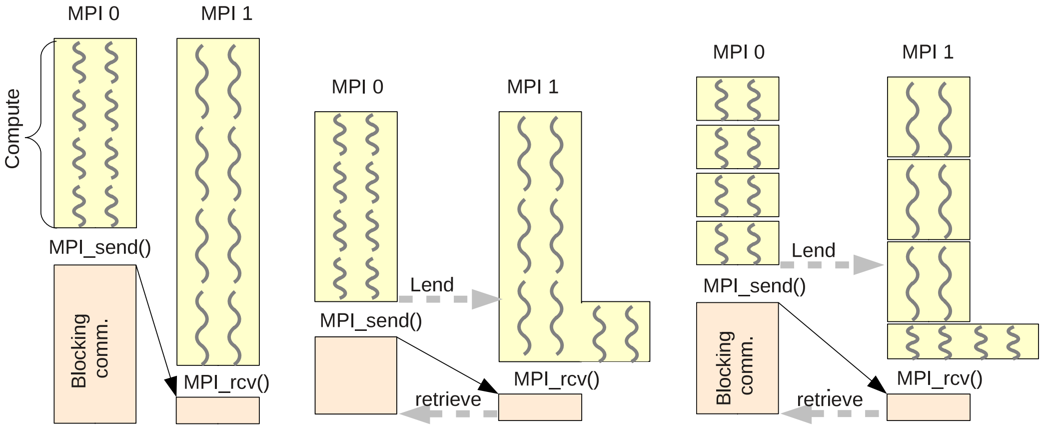
DLB (dynamic load balancing)

- “*Runtime interposition to [...] intercept MPI calls*”
- Balance load on the inner level (OpenMP/SMPSs)
- Several load balancing algorithms

LeWI (Lend CPU when Idle)

- CPUs of rank in blocking MPI call are idle
- Lend CPUs to other ranks and recover them after MPI call completes

LeWI



(a) No load balancing. (b) LeWI algorithm with SMPSs. (c) LeWI algorithm with OpenMP.

Approach

- “*Extensive performance evaluation*”
- “*Modeling parallelization characteristics that limit the automatic load balancing potential*”
- “*Improving automatic load balancing*”

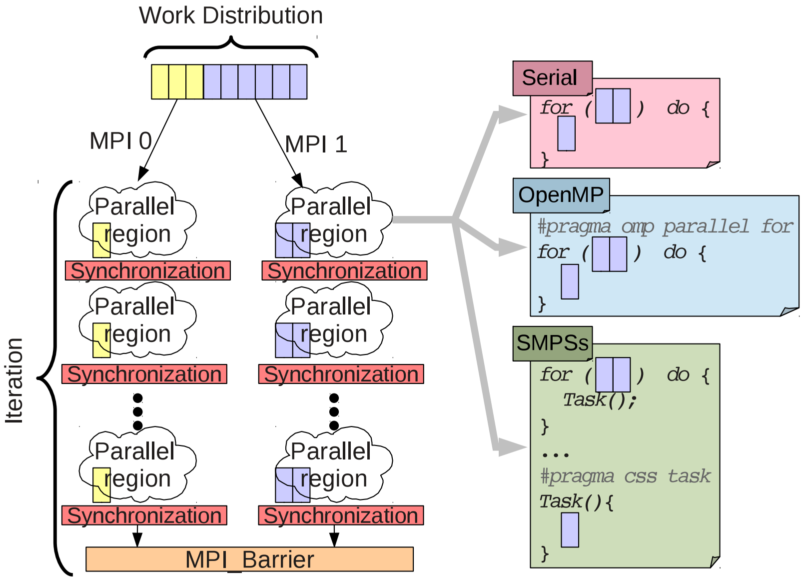
Performance evaluation

-
  - Marenostrum 2: 2 $\times$ IBM PowerPC 970MP (2 cores); 8 GiB RAM
  - Linux 2.6.5-7.244-pseries64; MPICH; IBM XL C/C++ compiler w/o optimizations
- Metrics
  - $Speedup = \frac{parallel\_execution\_time}{serial\_execution\_time}$
  - $Efficiency = \frac{useful\_cpu\_time}{elapsed\_time * cpus}$ where $useful\_cpu\_time = cpu\_time - (mpi\_time + openmp/smpss\_time + dlb\_time)$
  - $CPUs\_used$ to simultaneously run application code
- 3 benchmarks + 2 real applications

PILS (Parallel ImbaLance Simulation)

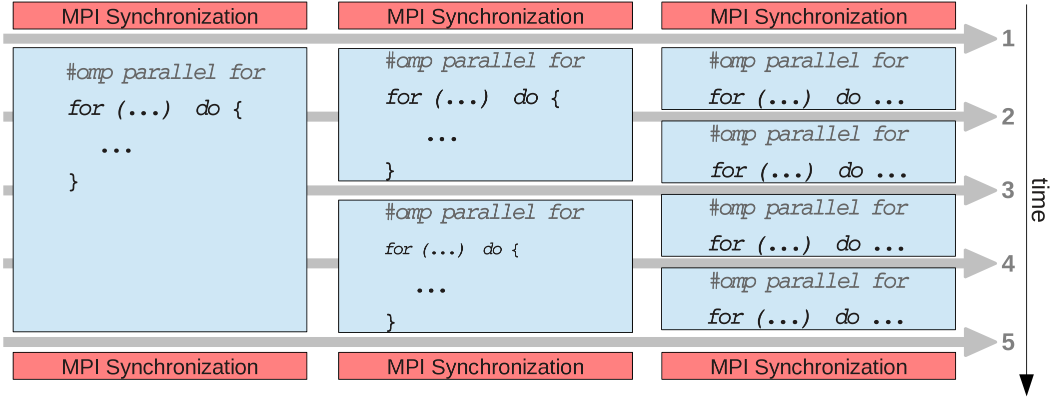
- Synthetic benchmark
- Core: "*floating point operations without data involved*"
- Tunable parameters
  - Programming model (MPI, MPI + OpenMP, MPI + SMPSs)
  - Load distribution
  - Parallelism grain ($= \frac{1}{\#parallel\ regions}$)
  - Iterations

PILS



Work Distribution
MPI 0
MPI 1
Iteration
Parallel region
Synchronization
Parallel region
Synchronization
Parallel region
Synchronization
Parallel region
Synchronization
Parallel region
Synchronization
Parallel region
Synchronization
MPI_Barrier
Serial
for ( ) do {
}
OpenMP
#pragma omp parallel for
for ( ) do {
}
SMPSs
for ( ) do {
Task();
}
...
#pragma css task
Task(){
}

Parallelism Grain



a) Parallelism Grain: 1. b) Parallelism Grain: 0.5. c) Parallelism Grain: 0.25.

Other Codes

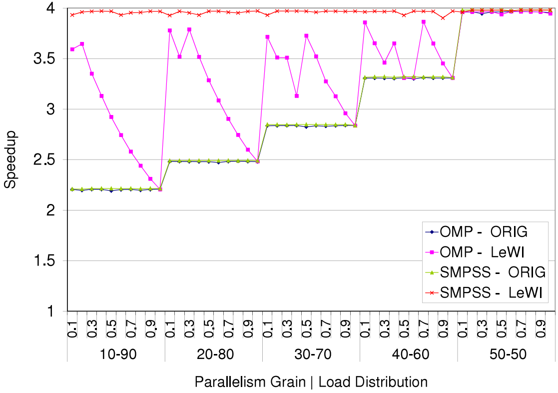
- Benchmarks
  - BT-MZ: block tri-diagonal solver
  - LUB: LU matrix factorization
- Applications
  - Gromacs: molecular dynamics, MPI-only
  - Gadget: cosmological N-body/SPH (smoothed-particle hydrodynamics) simulation

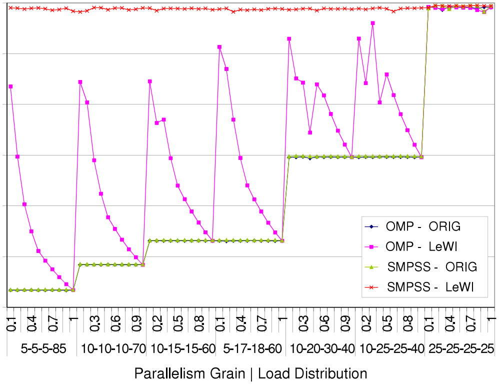
Other Codes

- Benchmarks
  - BT-MZ: block tri-diagonal solver
  - LUB: LU matrix factorization
- Applications
  - Gromacs: molecular dynamics, MPI-only
  - Gadget: cosmological N-body/SPH (smoothed-particle hydrodynamics) simulation

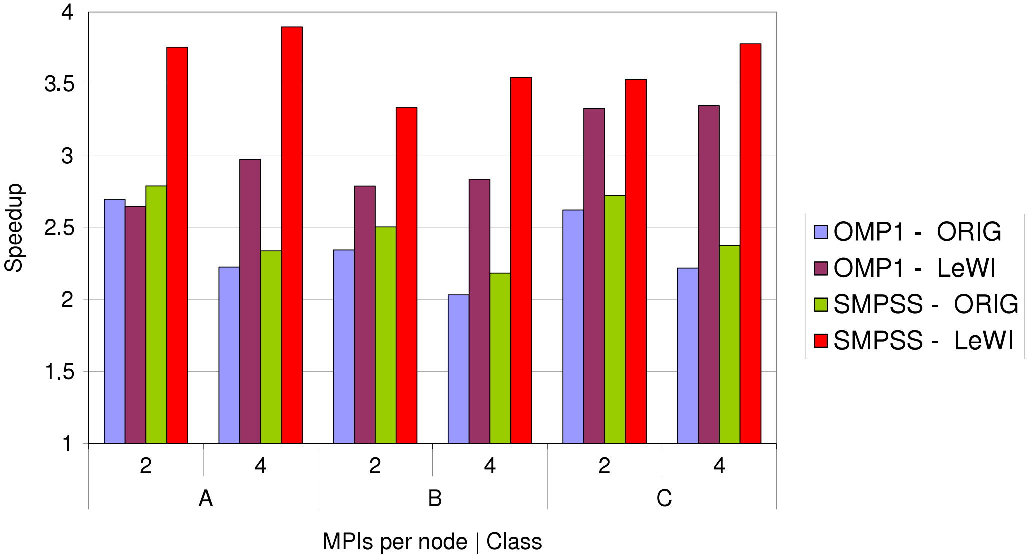
Application	Original version	MPI + OpenMP	MPI + SMPSs	Executed in nodes (cpus)
PILS	MPI + OpenMP MPI + SMPSs	X	X	1 (4)
BT-MZ	MPI + OpenMP	X	X	1, 2, 4 (4, 8, 16)
LUB	MPI + OpenMP MPI + SMPSs	X	X	1, 2, 4 (4, 8, 16)
Gromacs	MPI		X	1.64 (4.256)
Gadget	MPI	X		200 (800)

PILS, 2 and 4 MPI processes



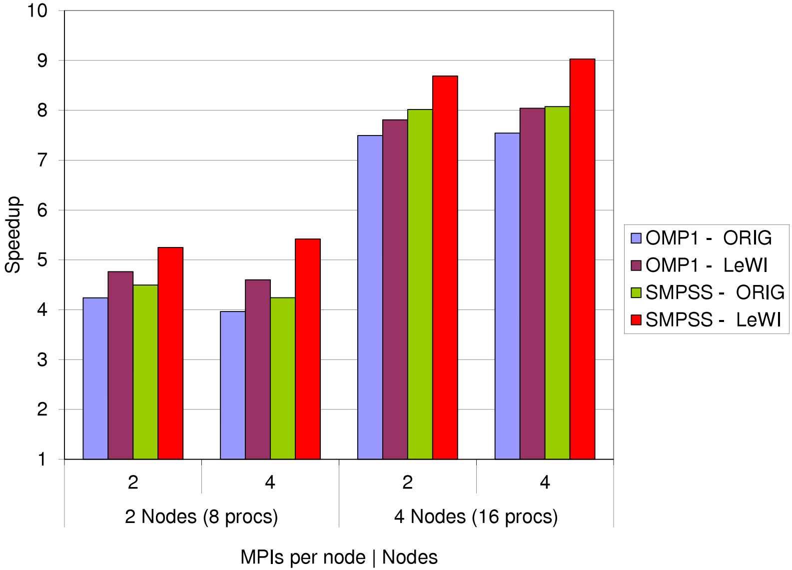


BT-MZ; 1 node

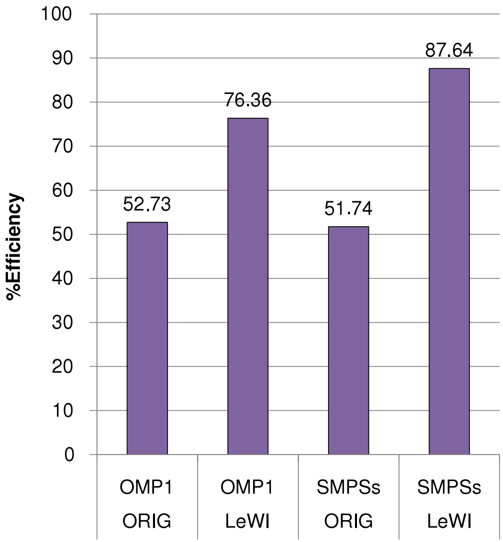


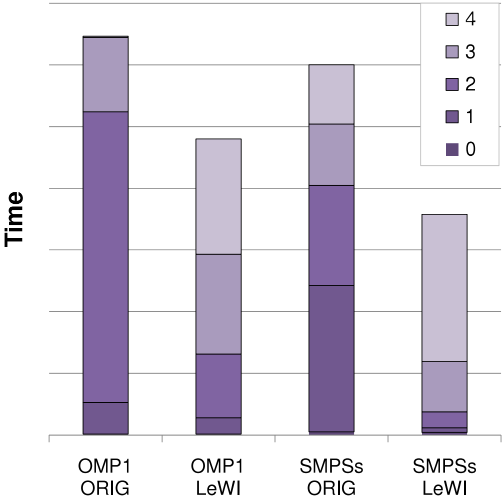
4
3.5
3
2.5
2
1.5
1
Speedup
2
4
2
4
2
4
A
B
C
MPIs per node | Class
OMP1 - ORIG
OMP1 - LeWI
SMPSS - ORIG
SMPSS - LeWI

BT-MZ; 2,4 nodes; Class C

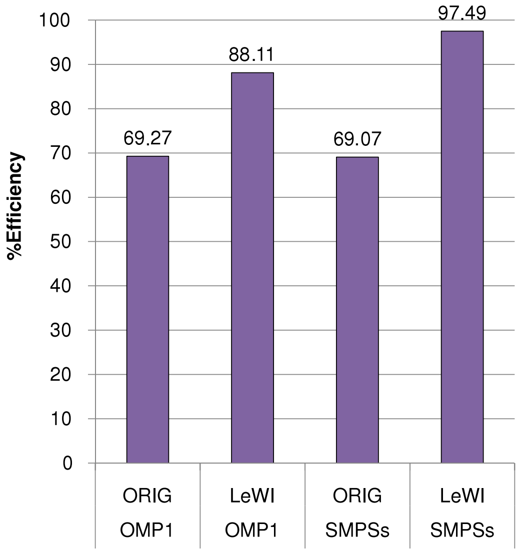


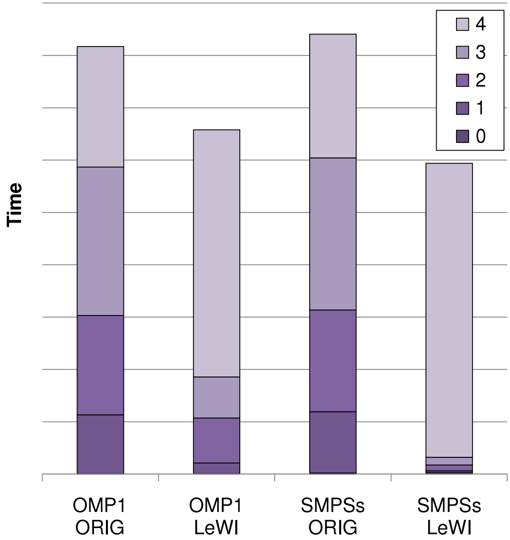
BT-MZ; 1 node; 4 MPI processes



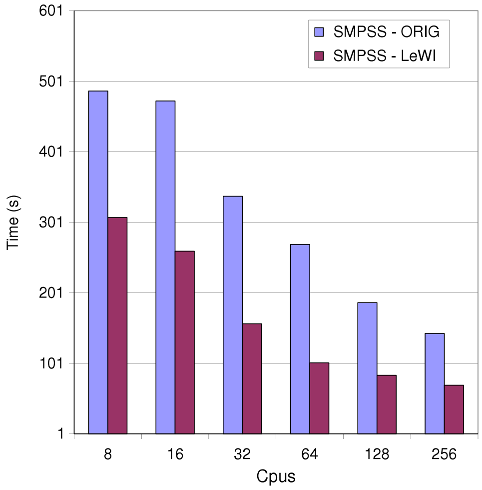


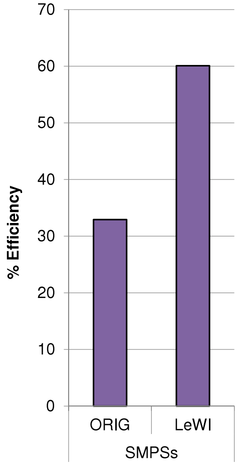
LUB; 1 node; Block size 200

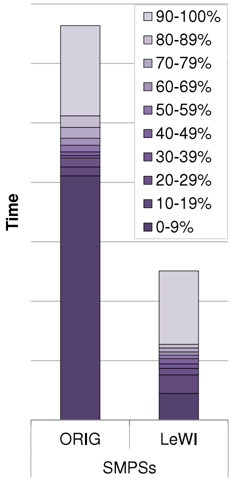




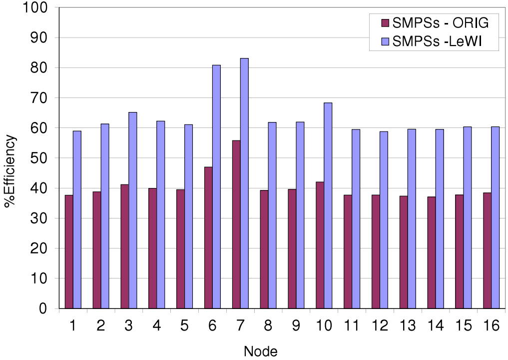
Gromacs; 1–64 nodes + Details for 16 nodes

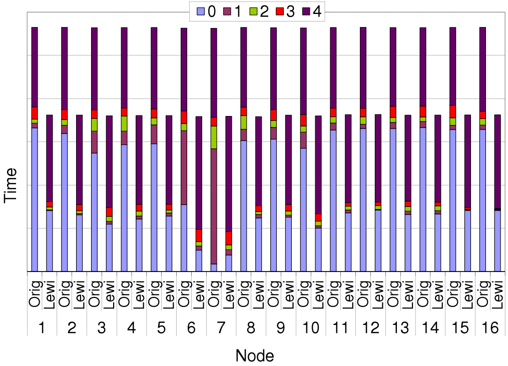




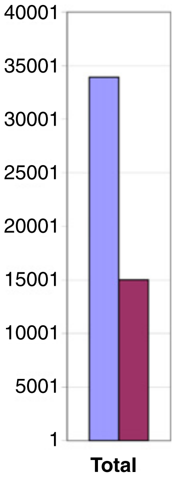


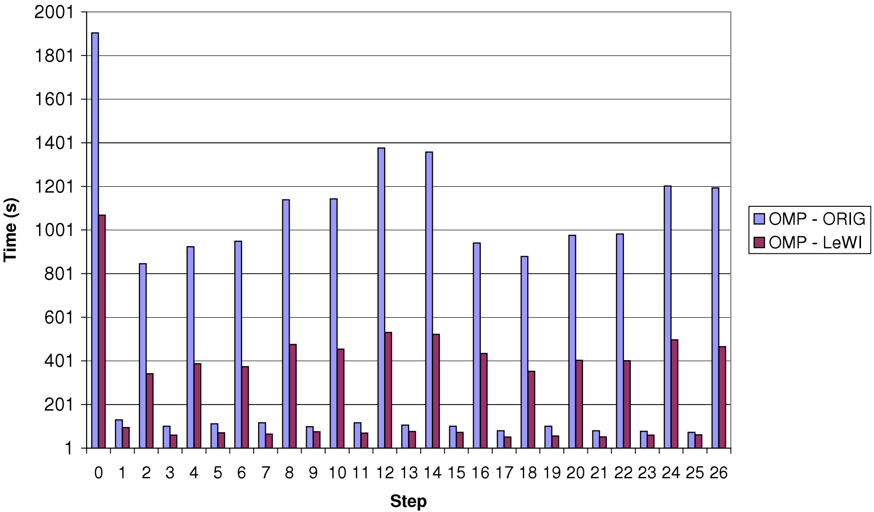
Gromacs; Efficiency + CPUs used per Node





Gadget; 200 nodes

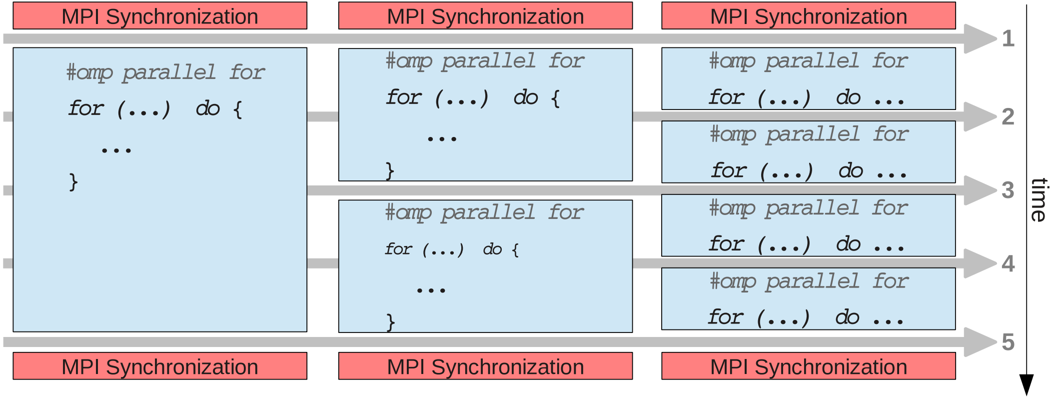




Factors Limiting Performance Improvement with LeWI

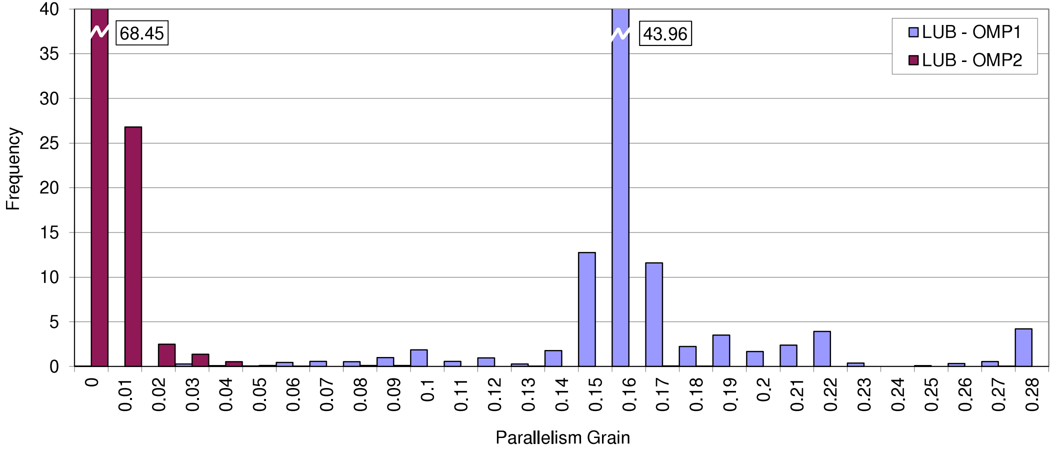
- "*Parallelism Grain in OpenMP applications*"
- "*Task duration in SMPSs applications*"
- "*Distribution of MPI processes among computation nodes*"

Parallelism Grain

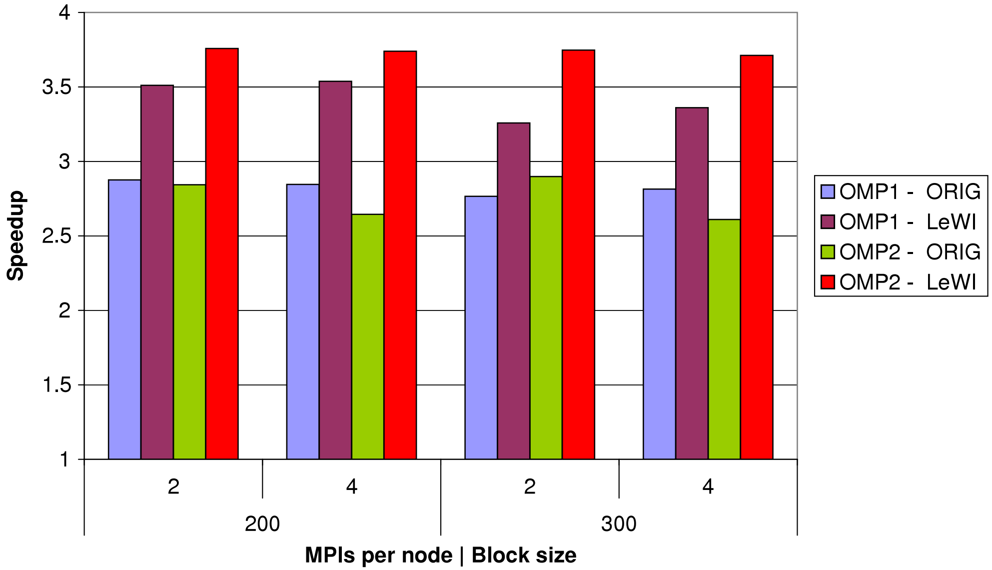


a) Parallelism Grain: 1. b) Parallelism Grain: 0.5. c) Parallelism Grain: 0.25.

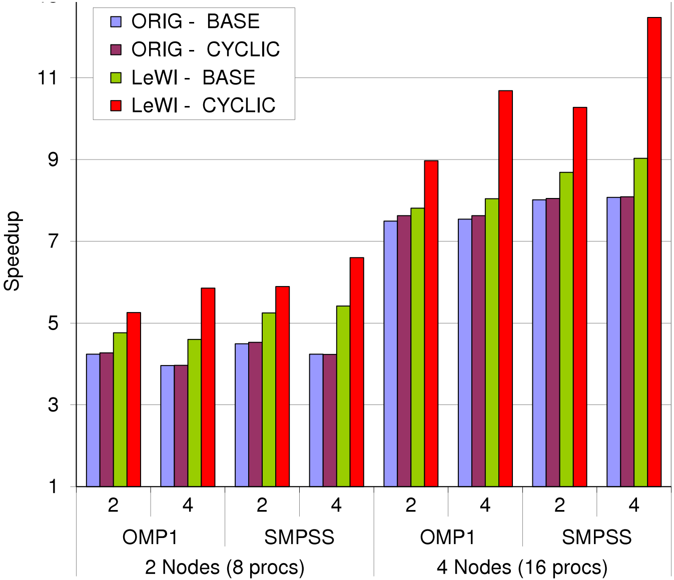
Modified Parallelism Grain in LUB



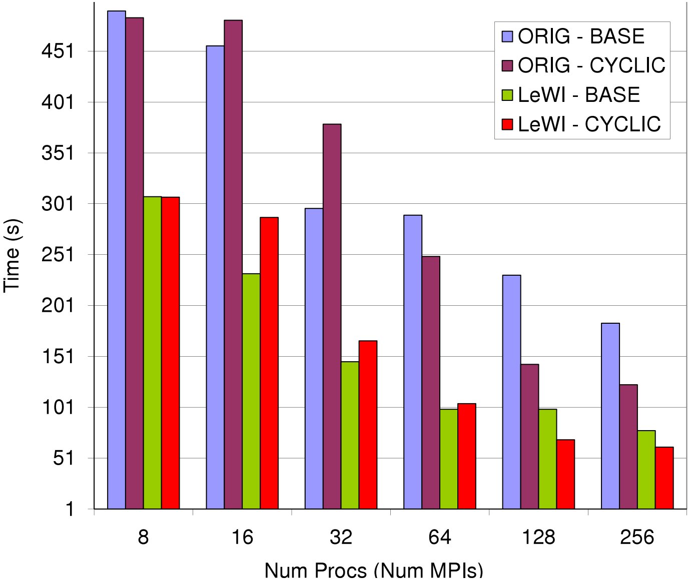
Performance of Modified LUB



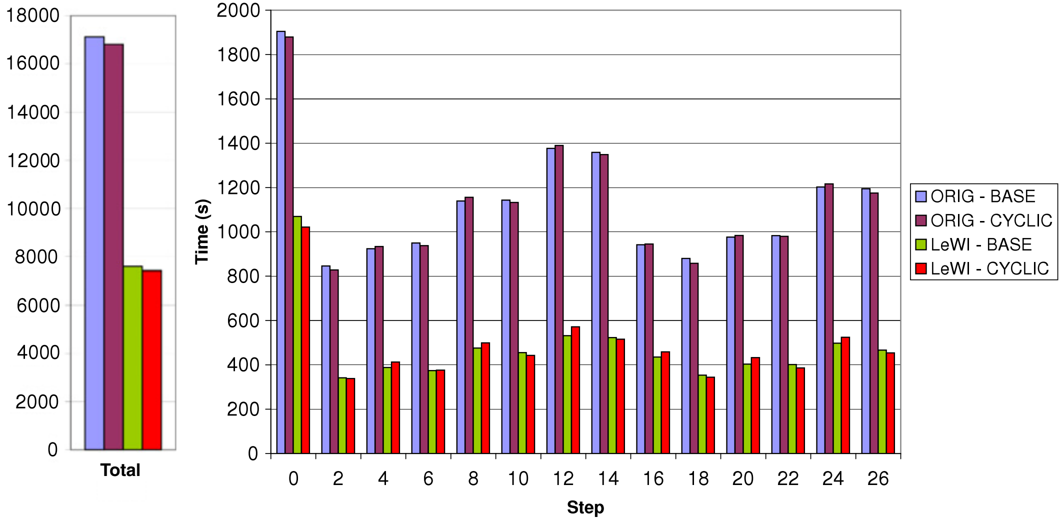
Rank Distribution — BT-MZ



Rank Distribution — Gromacs



Rank Distribution — Gadget



Conclusion

Summary

- DLB/LeWI can improve performance transparently
- Inter-node load imbalances not handled
- Granularity of parallelism and placement as important factors
- Optimal configuration with vs. without DLB/LeWI

Conclusion

Summary

- DLB/LeWI can improve performance transparently
- Inter-node load imbalances not handled
- Granularity of parallelism and placement as important factors
- Optimal configuration with vs. without DLB/LeWI

Discussion

- Interaction with MPI
- Benchmarks (1.5 of 3 NPB-MZ, arbitrary load distribution)
- How to find "the right" granularity