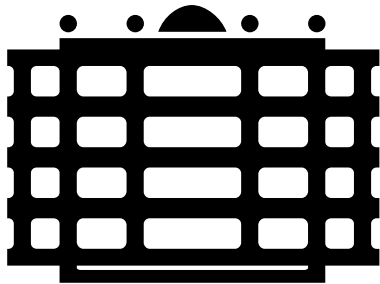


Interaction of Cache, Communication, and Load Increase on SMP Clusters for Parallel Adaptive FEM

Judith Hippold*, Gudula Rünger



**Chemnitz University of Technology
Department of Computer Science
09107 Chemnitz, Germany**

{ruenger, juh}@informatik.tu-chemnitz.de

* Supported by DFG, SFB393: Numerical Simulation on Massively Parallel Computers

Outline

- Motivation
- Parallel Platforms
- Application Program Characteristics
- Quantifying the Dependences
- Conclusion

Motivation

Adaptive FEM on distributed memory:

- strong input dependence
- unpredictable data access behavior
- load imbalances
- irregular communication behavior

Trends in parallel architectures:

- clusters of SMPs
- parallelism for a reasonable price
- pure MPI programs vs. mixed memory organization

- HW and SW characteristics form a **complex set of** dynamically interacting **dependences**
- Investigations to achieve efficient execution

Platforms

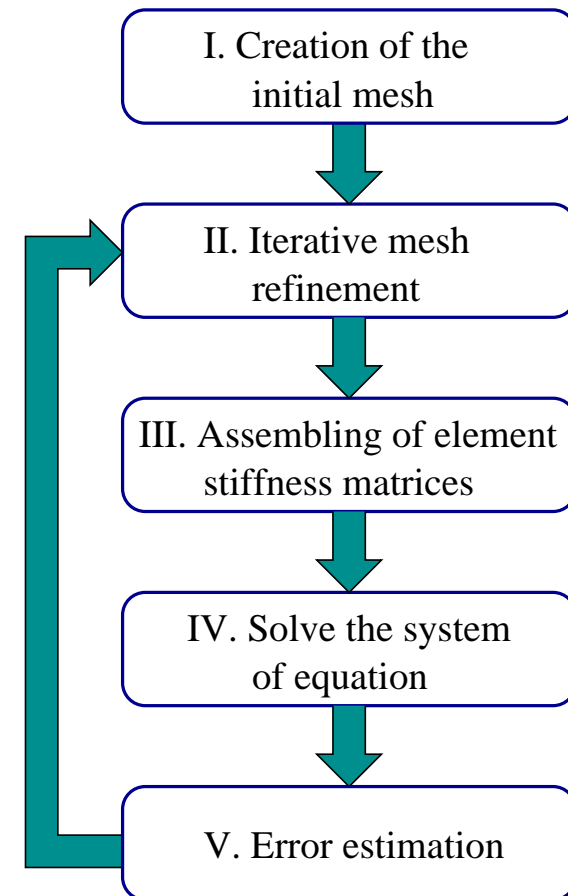
XEON: 16 PCs with 2 Intel Xeon processors (2.0 GHz);
SCI, ScaMPI;
8 KB L1-cache, 512 KB L2-cache, MESI;

SB1000: 4 SunBlade 1000 with 2 UltraSPARC3 processors (750 MHz);
SCI, ScaMPI;
Cache: 32 KB (I) / 64 KB (D) L1 (4-way set associative, 32 byte line size),
8 MB L2, MOESI;

JUMP: 41 Nodes with 32 Power4+ processors (1.7 GHz);
HPS, mpcc;
Cache: 64 KB (I) / 32 KB (D) L1 (internal), 1.5 MB L2 (per chip = 2 CPUs), 512 MB L3
(per frame = 32 CPUs);

Parallel Application

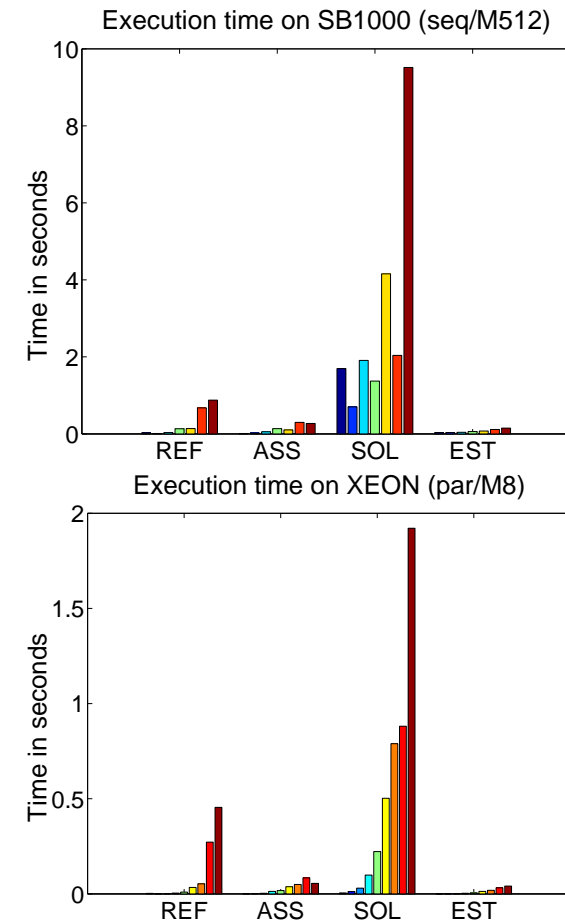
- adaptive, 3-dimensional, hexahedral finite elements, 8 or 20 or 27 nodes per element
- 2nd order elliptic partial differential problems (Poisson equation, Lamé system of linear elasticity)
- hierarchy of data structures (volumes, faces, edges, nodes)
- volumes are distributed among processors, duplicated structures
- element stiffness matrices and distributed vectors



Runtime Behavior

Performance determining factors:

- **Data structure size:** growing vector lengths, growing data structure lists
- **Communication effort:** reduced and optimized by algorithmic design & implementation strategies
- **Load balancing:** necessary due to AMR, time consuming because of additional communication and computation effort



Cache Dependences

Data structure sizes:

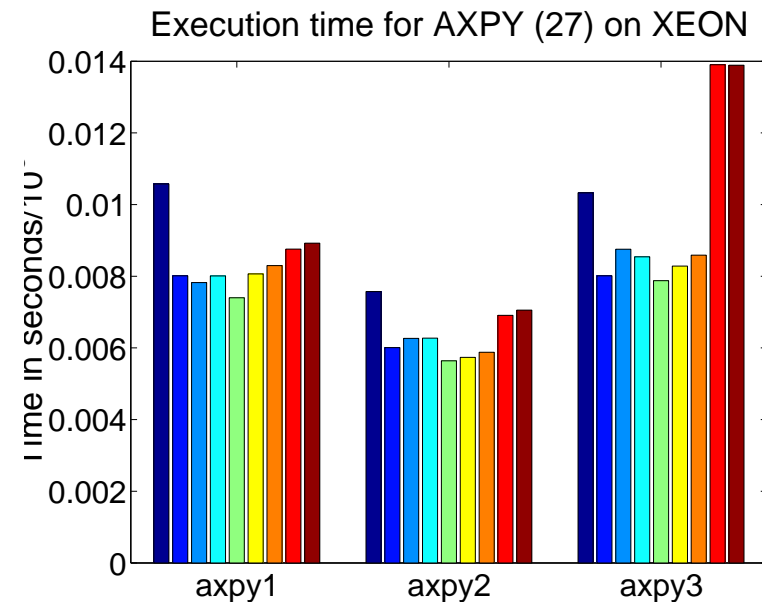
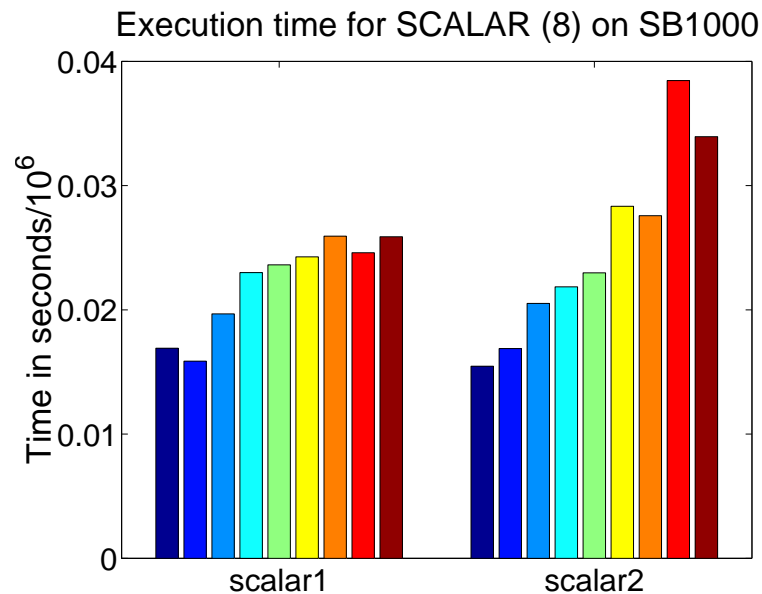
- growing solution vector lengths with increasing mesh refinement: e. g. 90 — 147 — 288 — 810 — 1500 — 3021 — 4212 — 7164 — 9054 entries
- constant element stiffness matrix size: $FE_nodes * degrees_of_freedom$
- depends on finite element type: e. g. 2.3 KB / 26 KB for 8 / 27 nodes per element

Access behavior:

- investigate sequential execution of time-consuming/frequent operations on large data structures
- AXPY: vector-vector-addition
- SCALAR: vector-vector-multiplication
- AXMEBE: matrix-vector-multiplication

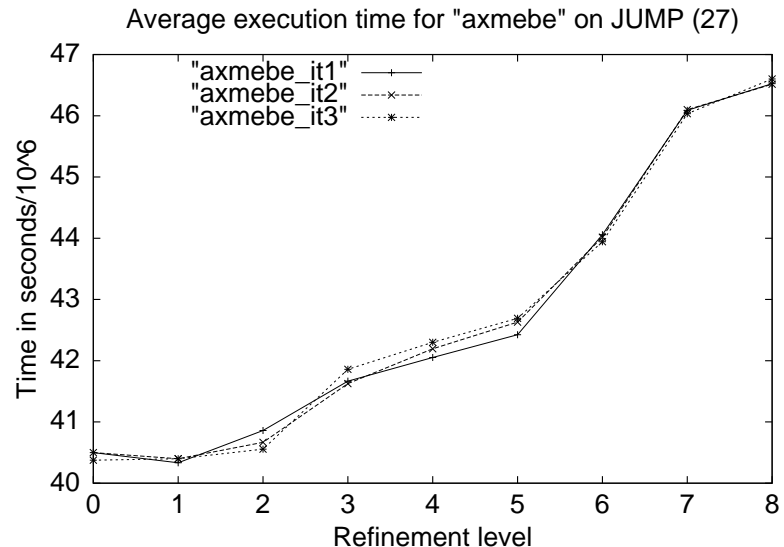
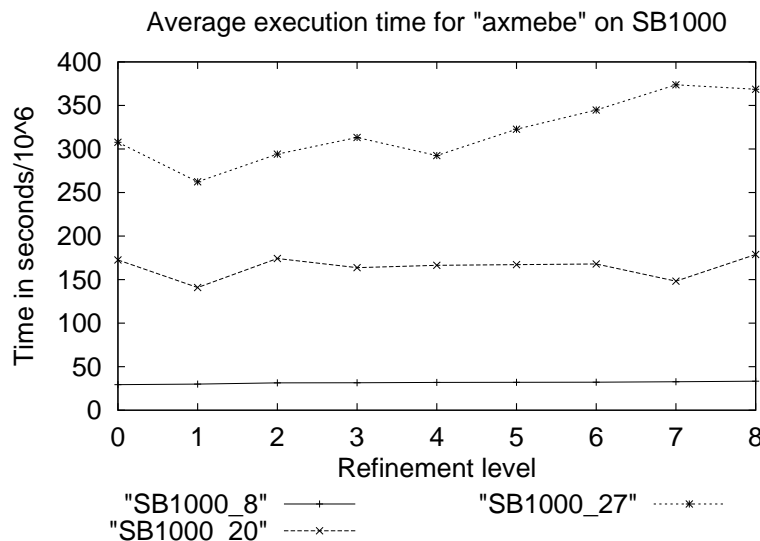
Operations SCALAR / AXPY

- SCALAR: $scalar = \underline{x} * \underline{y}$ AXPY: $\underline{y} = alpha * \underline{x} + \underline{y}$
- regular accesses
- \underline{x} and \underline{y} are global vectors (length: $total_nodes * degrees_of_freedom$)
- growing vector lengths with increasing refinement
- performed several times, average time to calculate one entry



Operation AXMEBE

- $\underline{y} = \underline{y} + A * \underline{u}$
- A is the element stiffness matrix (triangular, length of the diagonal: $FE_nodes * degrees_of_freedom$)
- \underline{y} and \underline{u} are global vectors (length: $total_nodes * degrees_of_freedom$), irregular accesses
- usage of auxiliary vectors (length: $FE_nodes * degrees_of_freedom$)



Time Consumption

It.	SB1000			XEON			JUMP		
	axmebe	scalar	axpy	axmebe	scalar	axpy	axmebe	scalar	axpy
1	30.05	0.36	0.63	30.71	1.18	1.90	76.25	6.60	9.09
3	78.49	0.79	1.23	34.11	0.78	0.47	77.27	2.43	2.83
5	80.61	0.69	1.00	37.73	0.30	0.52	80.18	1.16	1.48
7	81.33	0.67	1.00	37.84	0.42	0.27	74.51	0.87	1.28
9	79.81	0.69	1.12	38.48	0.26	0.39	72.52	0.81	1.14

Time consumed by the different operations in relation to the total time needed for solving the system of equations using 8 nodes per finite element.

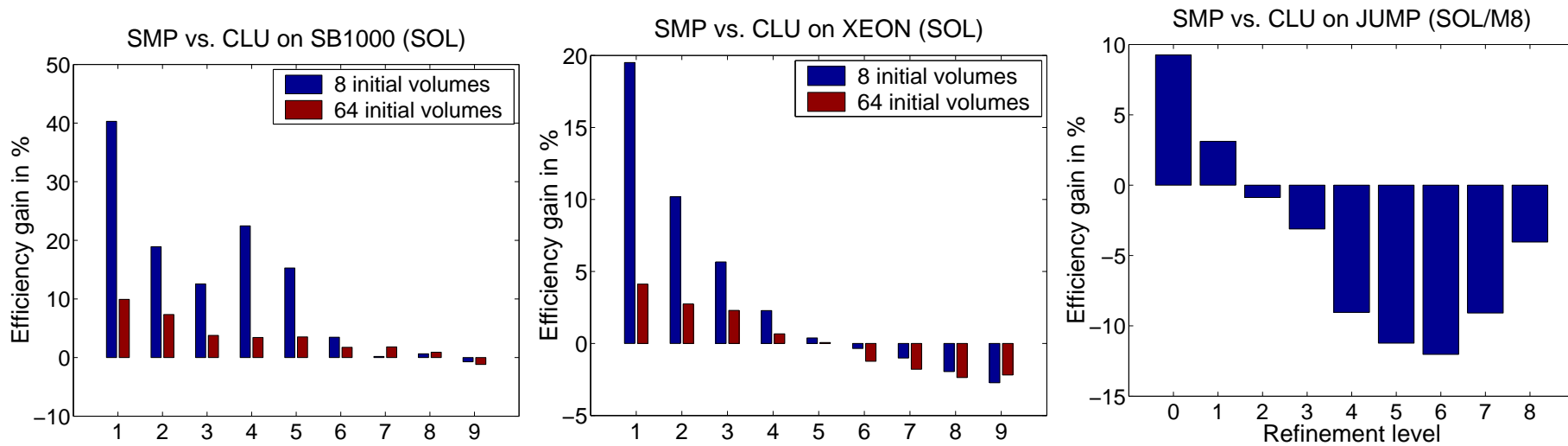
Communication needs

- global exchanges (GE) = all processors exchange gathered data

Adaptive refinement (REF):	$refinement_iterations * (2GE)$
Assemble element stiffness matrices (ASS):	$(1GE)$
Solve the system of equations (SOL):	$solver_iterations * (6GE) + (2GE)$
Error estimation (EST):	$(2GE)$

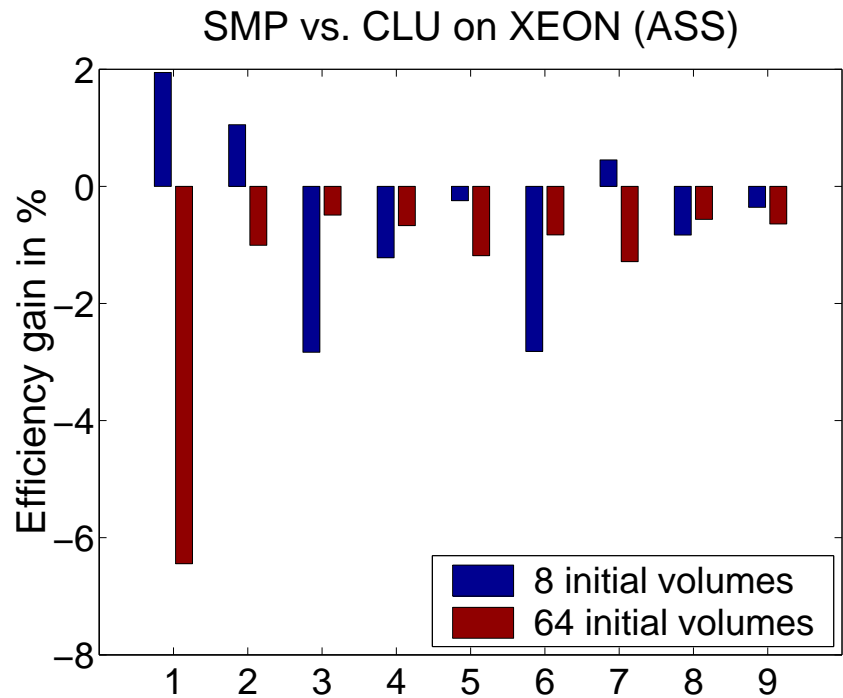
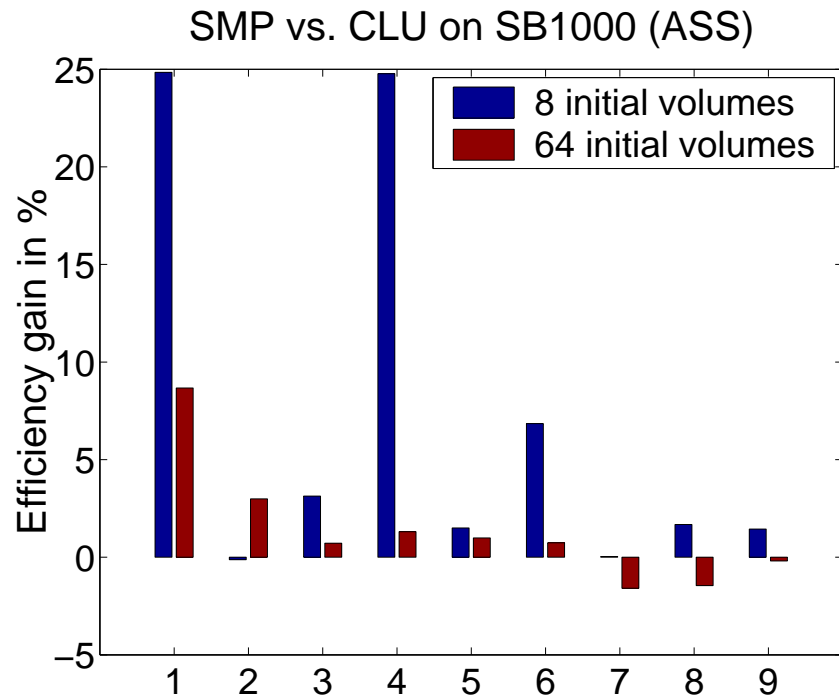
- most communication effort during solving the system of equations
- different process assignment strategies: **to the same cluster node** (SMP) or **to different cluster nodes** (CLU)
- Influences on program execution time?

Comparison (1)



- Comparison of the two assignment strategies for solving the system of equations
- different initial number of finite elements
- communication intensive
- positive scale: SMP version, negative scale: CLU version benefits

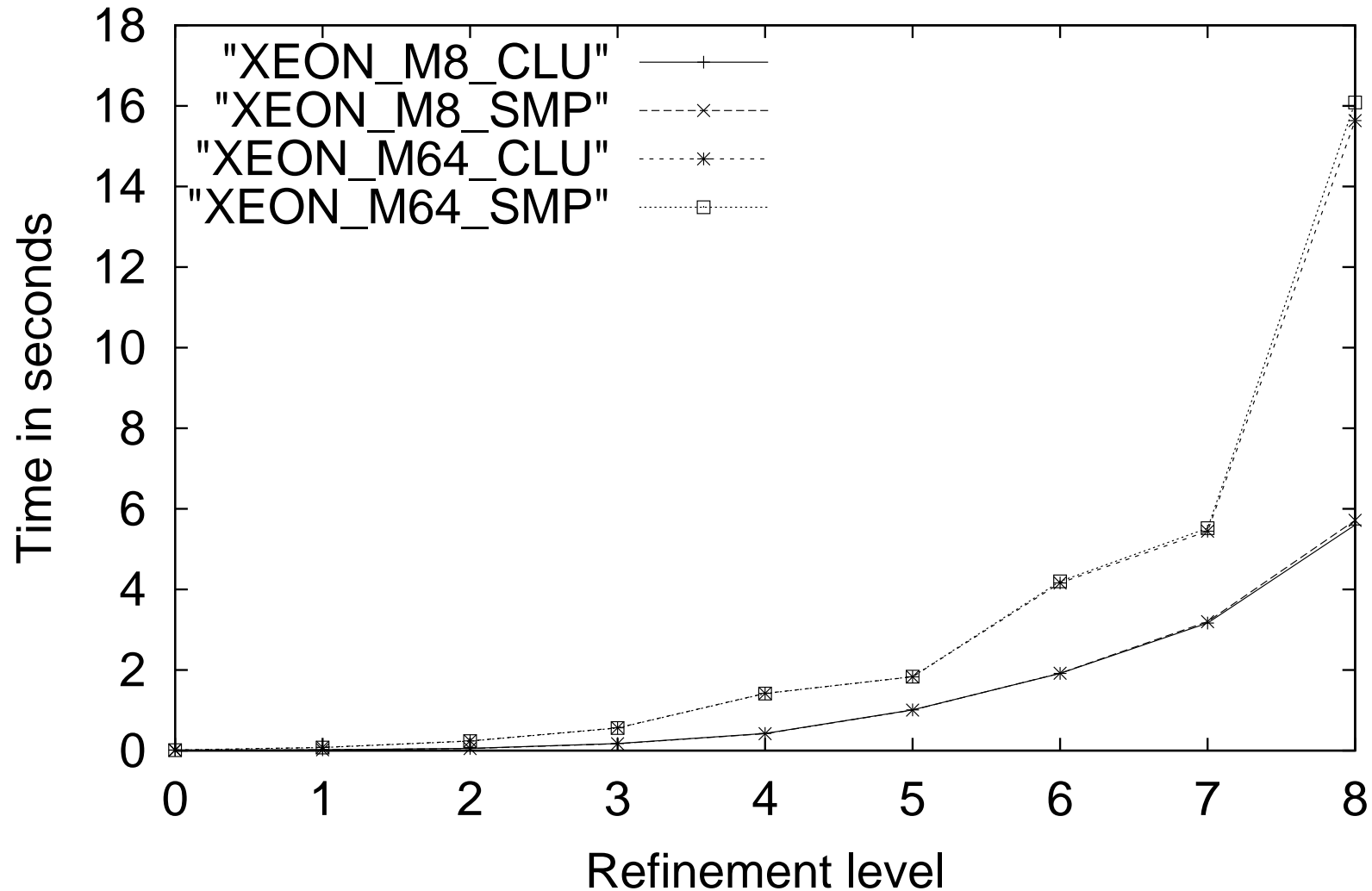
Comparison (2)



Comparison of the two assignment strategies for assembling the element stiffness matrices (different initial number of finite elements, 1 GE).

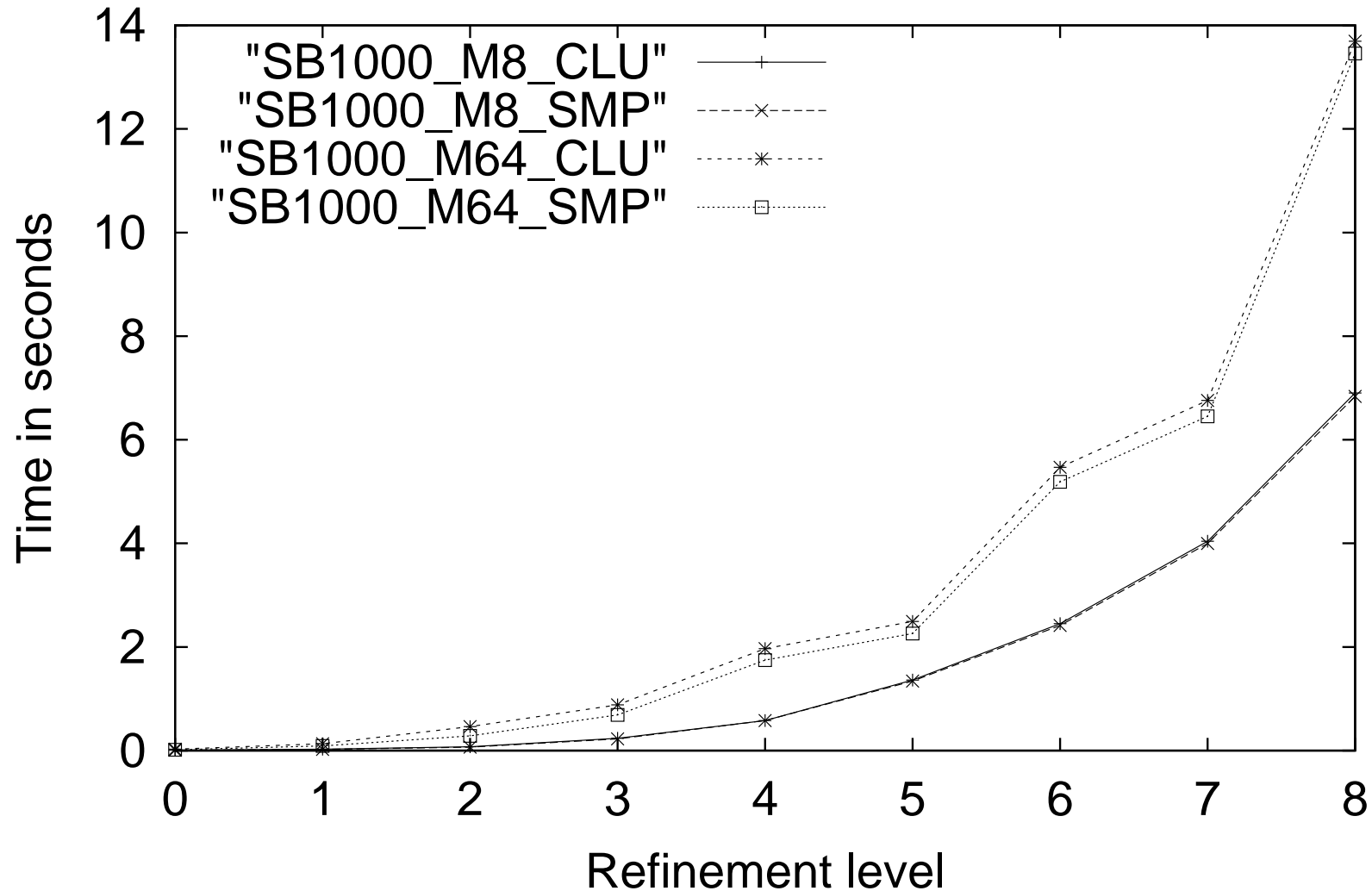
Total Runtime (XEON)

Runtime on two processors of XEON



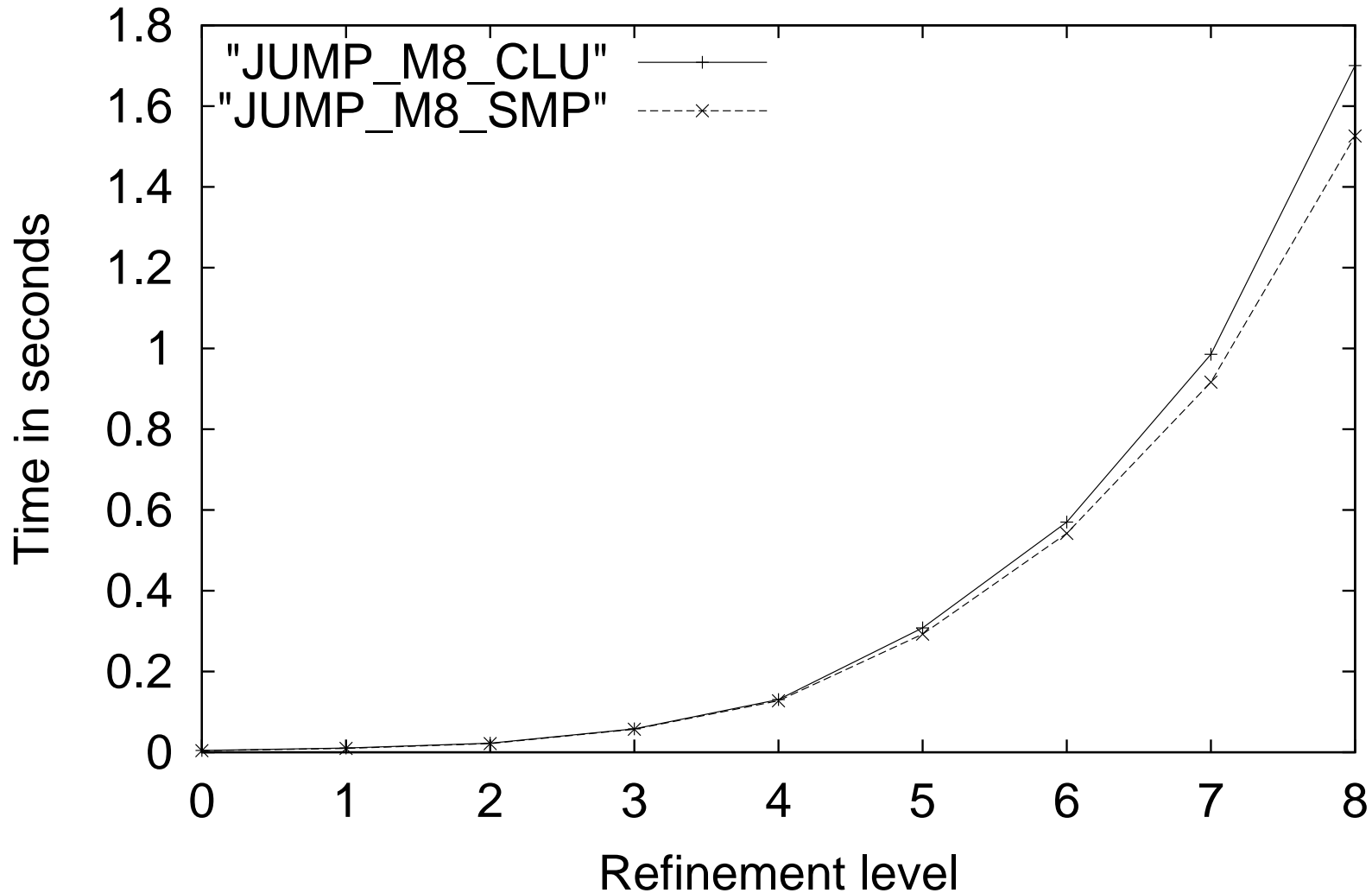
Total Runtime (SB1000)

Runtime on two processors of SB1000



Total Runtime (JUMP)

Runtime on two processors of JUMP



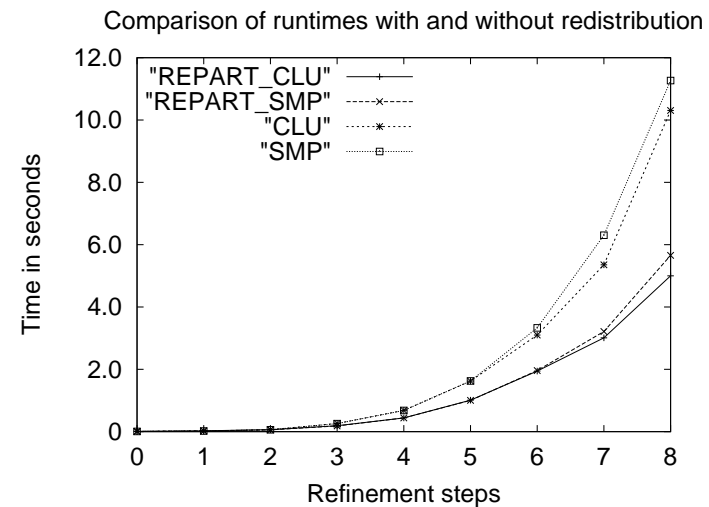
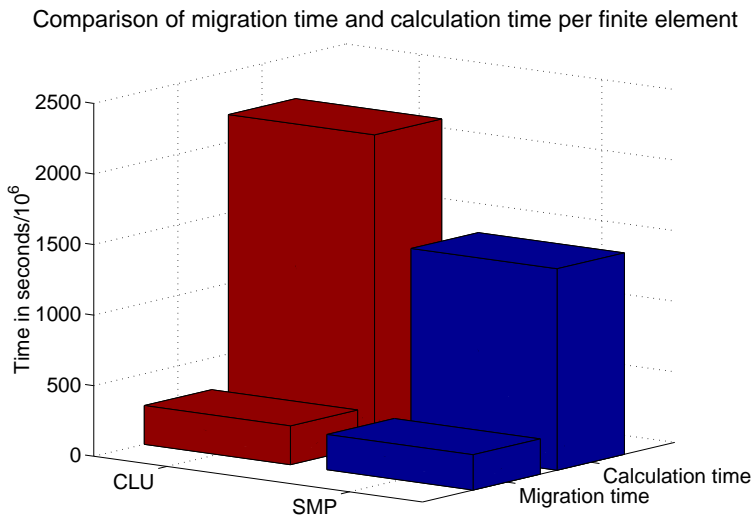
Conclusion

- data structure distribution offers good scalability (decreases vector lengths and data structure list lengths)
- complex and interweaved impacts of HW and program characteristics on execution time

Optimization approaches:

- dynamic process creation: trade-off between the number of finite elements to compute and the resulting communication overhead
- platform specific process assignment strategy

- dynamic repartitioning strategy - first approach: comparison of migration time and computation time during program run



Thank you for your attention.

