CARP-CG: a computational study

Art Petrenko, December 4, 2012

SLIM
University of British Columbia

Motivation

- Full waveform inversion requires many forward solves: $\mathcal{O}(n_f n_s n_{\text{iter}})$
- Since the matrix is very large ($n_x n_y n_z$), direct solvers use too much memory.
- The goal: accelerate an iterative solver.

SLIM

Outline

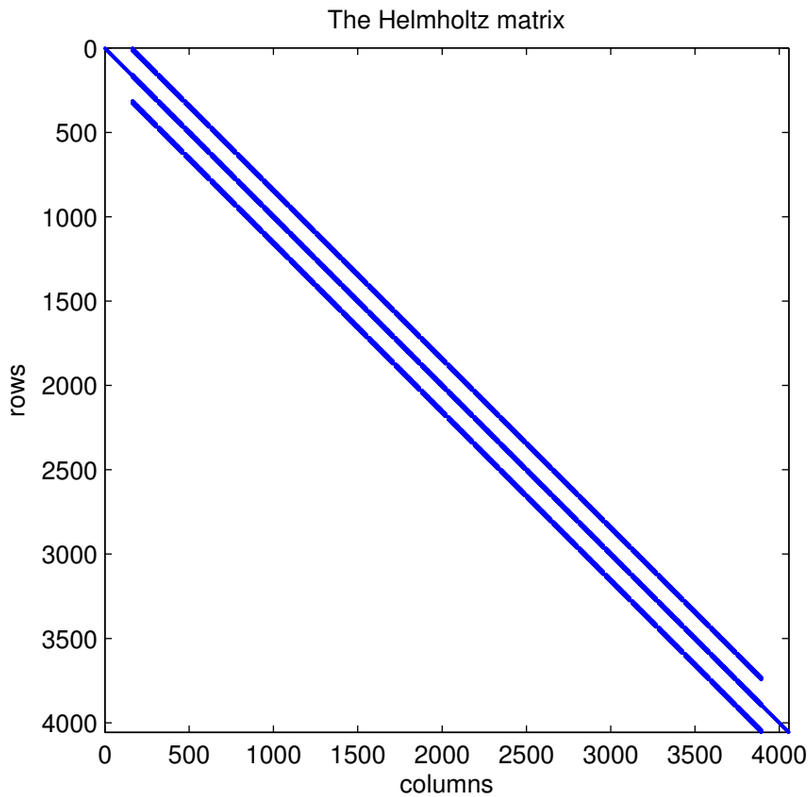
- Problem: solving a linear system
- Algorithm: CARP and CG
- Acceleration: multi-threading
- Performance
- Conclusions

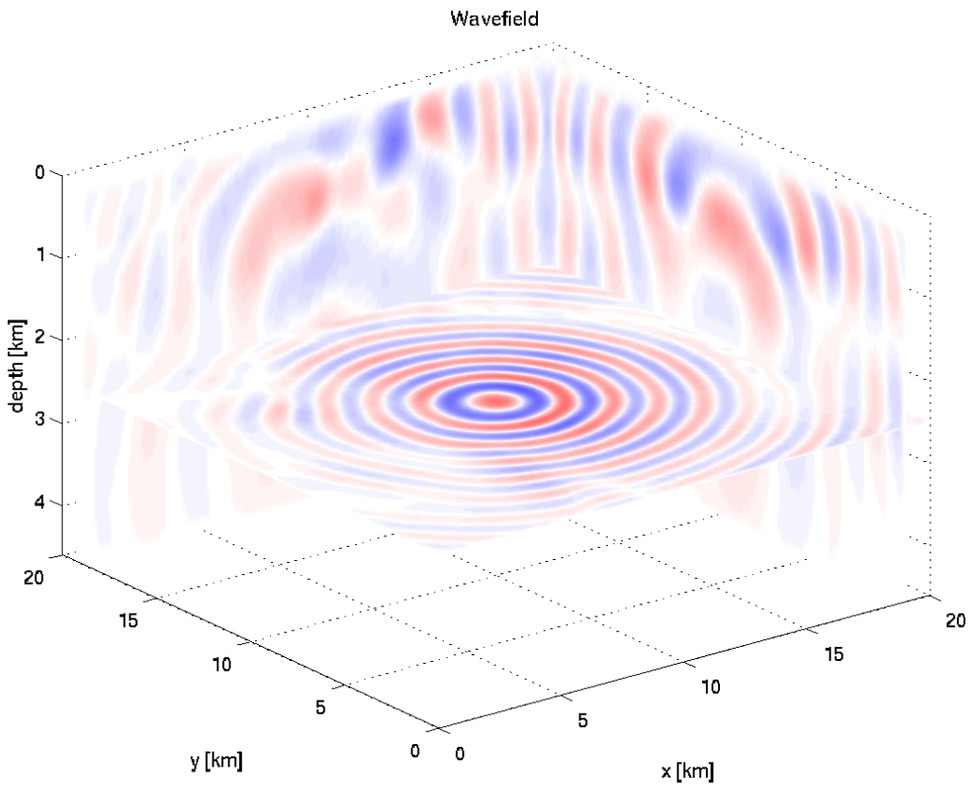
The Helmholtz system

- We solve the constant density (ρ = 1) isotropic acoustic wave equation in the frequency domain:

$$\left(\omega^2\mathbf{m} + \frac{1}{\xi_x(x)}\frac{\partial}{\partial x}\frac{1}{\xi_x(x)}\frac{\partial}{\partial x} + \frac{1}{\xi_y(y)}\frac{\partial}{\partial y}\frac{1}{\xi_y(y)}\frac{\partial}{\partial y} + \frac{1}{\xi_z(z)}\frac{\partial}{\partial z}\frac{1}{\xi_z(z)}\frac{\partial}{\partial z}\right)\mathbf{pressure} = \mathbf{source}$$

$$H\mathbf{u} = \mathbf{q}$$



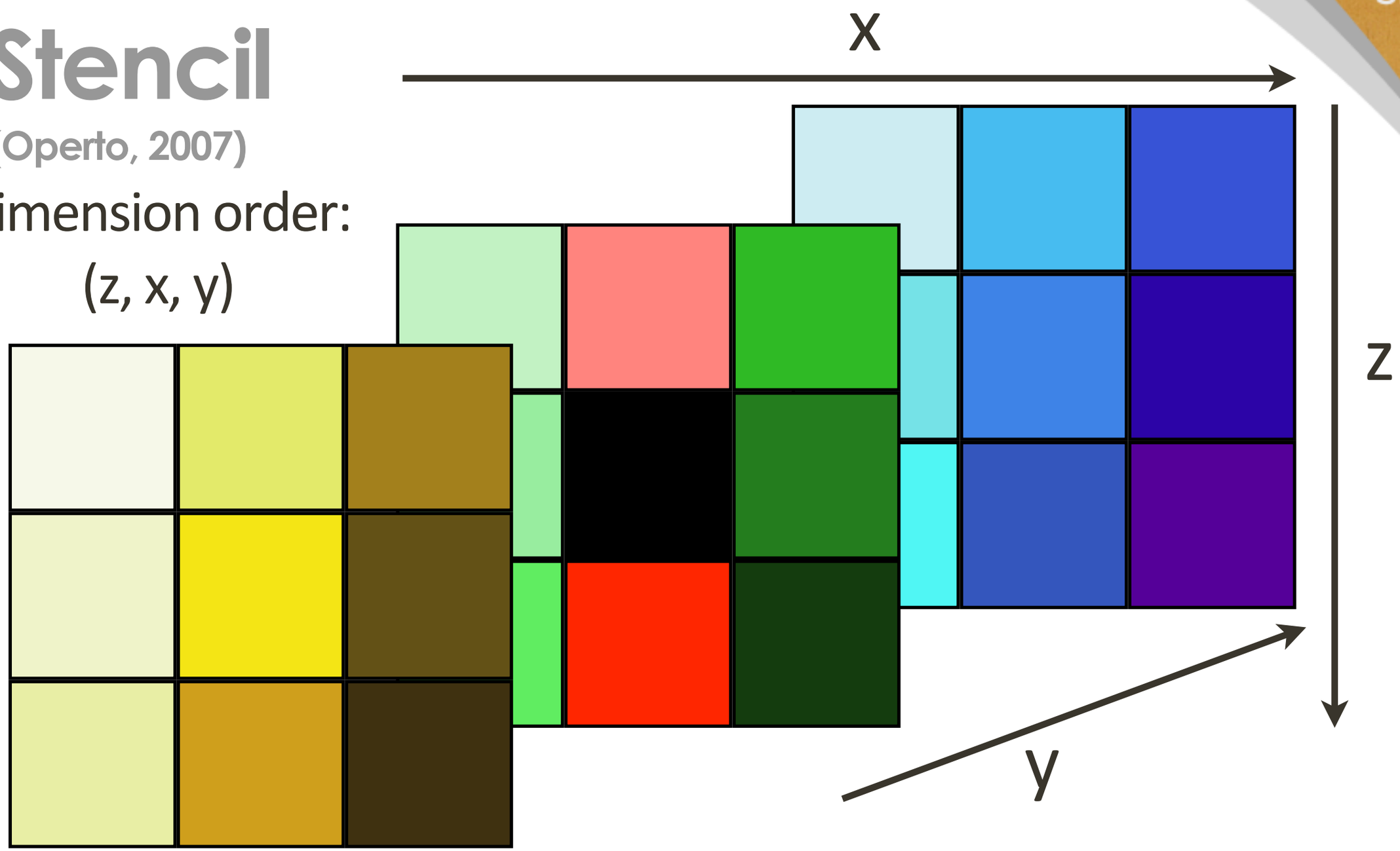


Stencil

(Operto, 2007)

dimension order:

(z, x, y)



Preconditioning needed

- H is not positive definite, hence the method of conjugate gradients is not suitable to solve

$$H\mathbf{u} = \mathbf{q}$$

- We precondition the system into an equivalent, positive definite form:

$$(I - Q)\mathbf{u} = R\mathbf{q}$$

$$A\mathbf{u} = \mathbf{b}$$

- Instead of forming the matrices explicitly, we use the ***double CARP sweep*** operator (Gordon & Gordon, 2005):

$$\mathrm{DCSWP}(H, \mathbf{u}, \mathbf{q}, w) = Q\mathbf{u} + R\mathbf{q}$$

where w is a relaxation parameter.

CARP sweeps explained

- An iterative method for solving generic linear systems.
- The current iterate is projected onto the plane defined by a row of the matrix to become the next iterate:

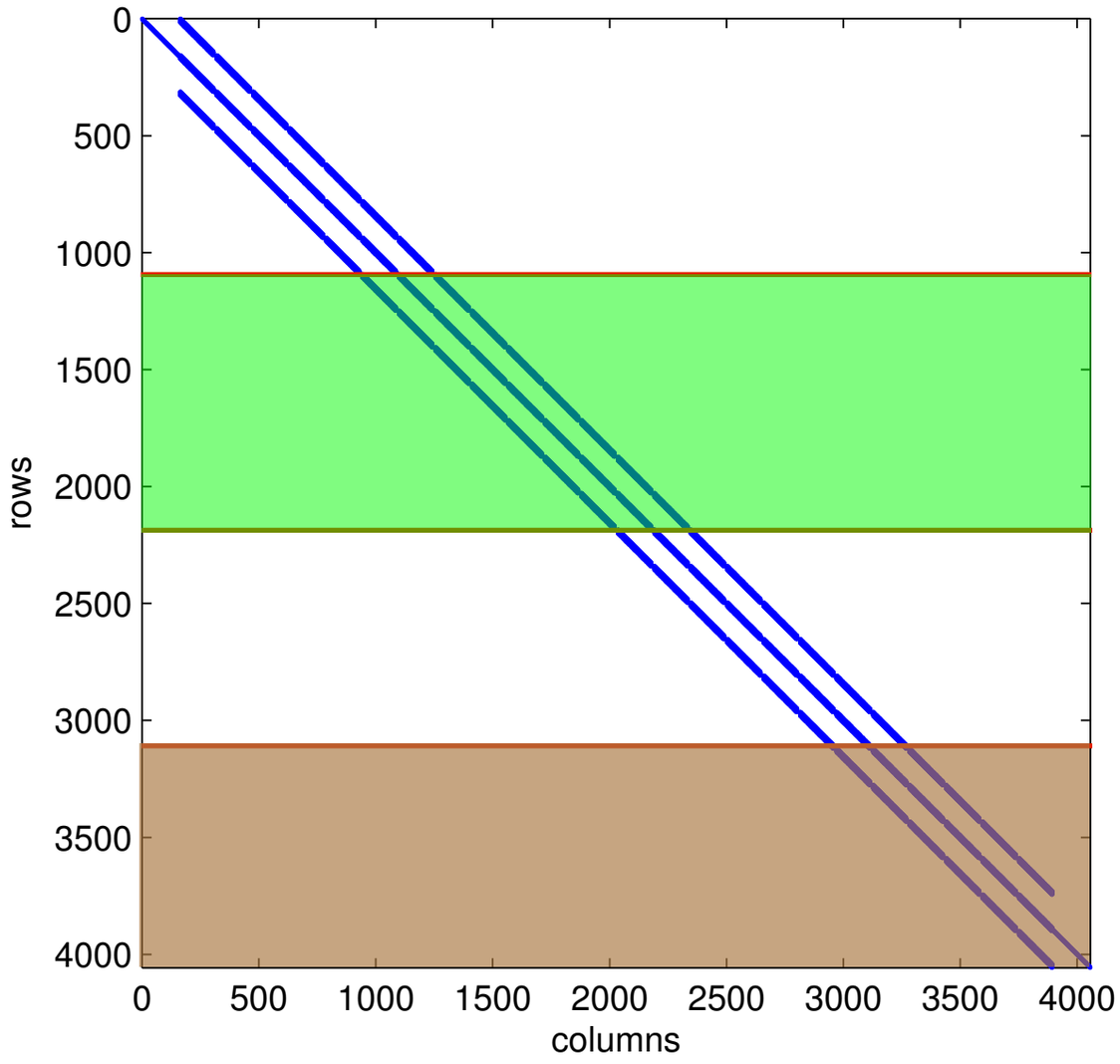
$$\mathbf{u}^{i+1} = \mathbf{u}^i + w(q_k - \mathbf{a}_k \mathbf{u}^i)\mathbf{a}_k^*$$

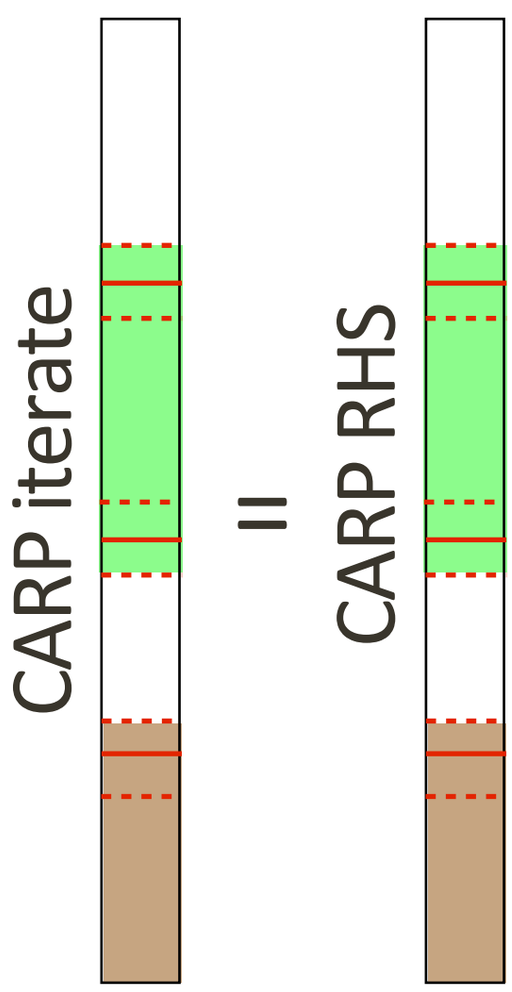
- The matrix rows ($\mathbf{a}_k$) are divided into possibly overlapping sets which undergo the algorithm in ***parallel.***
- Iterate ($\mathbf{u}^i$) elements shared between sets are averaged.
- Index during double sweep:

$$k : 1 \rightarrow N \rightarrow 1$$

Parallelization

The Helmholtz matrix





nodes or threads

The CARP-CG algorithm

(Gordon & Gordon, 2010)

Vanilla CG

Input: H, $\mathbf{u^0}$, $\mathbf{q}$

$i = 0$

$\mathbf{r}^0 = \mathbf{q} - H\mathbf{u}^0$

$\mathbf{p}^0 = \mathbf{r}^0$

while $\left\|\mathbf{r}^i\right\|^2 > tol$ **do**

$\mathbf{q}^i_{cg} = H\mathbf{p}^i$

$\alpha = \left\|\mathbf{r}^i\right\|^2 / \left\langle \mathbf{p}^i, \mathbf{q}^i_{cg} \right\rangle$

$\mathbf{u}^{i+1} = \mathbf{u}^i + \alpha \mathbf{p}^i$

$\mathbf{r}^{i+1} = \mathbf{r}^i - \alpha \mathbf{q}^i_{cg}$

$\beta = \left\|\mathbf{r}^{i+1}\right\|^2 / \left\|\mathbf{r}^i\right\|^2$

$\mathbf{p}^{i+1} = \mathbf{r}^{i+1} + \beta \mathbf{p}^i$

$i = i + 1$

end while

Output: u

CARP-CG

Input: H, $\mathbf{u^0}$, $\mathbf{q}$, w

$i = 0$

$\mathbf{b} = \mathrm{DCSWP}(H, \mathbf{0}, \mathbf{q}^0, w)$

$\mathbf{r}^0 = \mathbf{b} - \mathbf{u}^0 + \mathrm{DCSWP}(H, \mathbf{u}^0, \mathbf{0}, w)$

$\mathbf{p}^0 = \mathbf{r}^0$

while $\left\|\mathbf{r}^i\right\|^2 > tol$ **do**

$\boldsymbol{tmp} = \mathrm{FCSWP}(H, \mathbf{p^i}, \mathbf{0}, w)$

$\mathbf{q}^i_{cg} = \mathbf{p}^i - \mathrm{BCSWP}(H, \boldsymbol{tmp}, \mathbf{0}, w)$

$\alpha = \left\|\mathbf{r}^i\right\|^2 / \left\langle \mathbf{p}^i, \mathbf{q}^i_{cg} \right\rangle$

$\mathbf{u}^{i+1} = \mathbf{u}^i + \alpha \mathbf{p}^i$

$\mathbf{r}^{i+1} = \mathbf{r}^i - \alpha \mathbf{q}^i_{cg}$

$\beta = \left\|\mathbf{r}^{i+1}\right\|^2 / \left\|\mathbf{r}^i\right\|^2$

$\mathbf{p}^{i+1} = \mathbf{r}^{i+1} + \beta \mathbf{p}^i$

$i = i + 1$

end while

Output: u

Implementation

- **CG:** MATLAB
- **CARP sweep:** C source compiled into MEX file that is called from MATLAB like a function.
- CARP source:

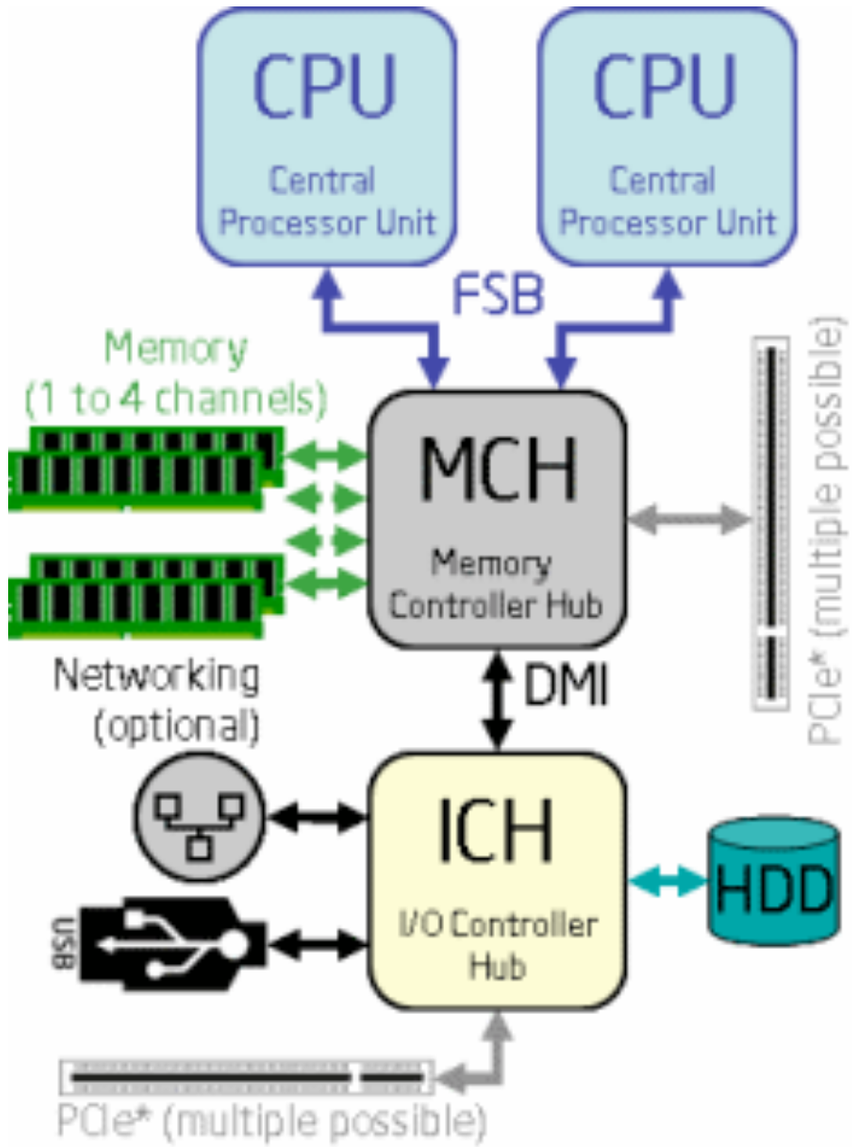
```
for(i=0;i<nrow;i++){
    cr = br[i+(int)idx[idx_mid]];
    ci = bi[i+(int)idx[idx_mid]];
    for(j=0;j<ncol;j++){
        k = i + (int)idx[j];
        if(k>=0 && k<nx){
            cr -= Sr[i*ncol + j]*yr[k] - Si[i*ncol + j]*yi[k];
            ci -= Sr[i*ncol + j]*yi[k] + Si[i*ncol + j]*yr[k];
        }
    }
    cr *= w;
    ci *= w;
    for(j=0;j<ncol;j++){
        k = i + (int)idx[j];
        if(k>=0 && k<nx){
            yr[k] +=   cr*Sr[i*ncol + j] + ci*Si[i*ncol + j];
            yi[k] +=  -cr*Si[i*ncol + j] + ci*Sr[i*ncol + j];
        }
    }
}
```

Implementation details

- Helmholtz matrix is sparse and structured; it is stored as rows of non-zero diagonals and their offsets.
- Row-wise storage of diagonals ensures memory is accessed sequentially.
- pthreads library used for multi-threading.
- gcc with the -O option used for compilation.
- Also tried -O2 -O3, difference of less than 5% in timing.
- Single-threaded FORTRAN sweeps implemented, but were slower than C.

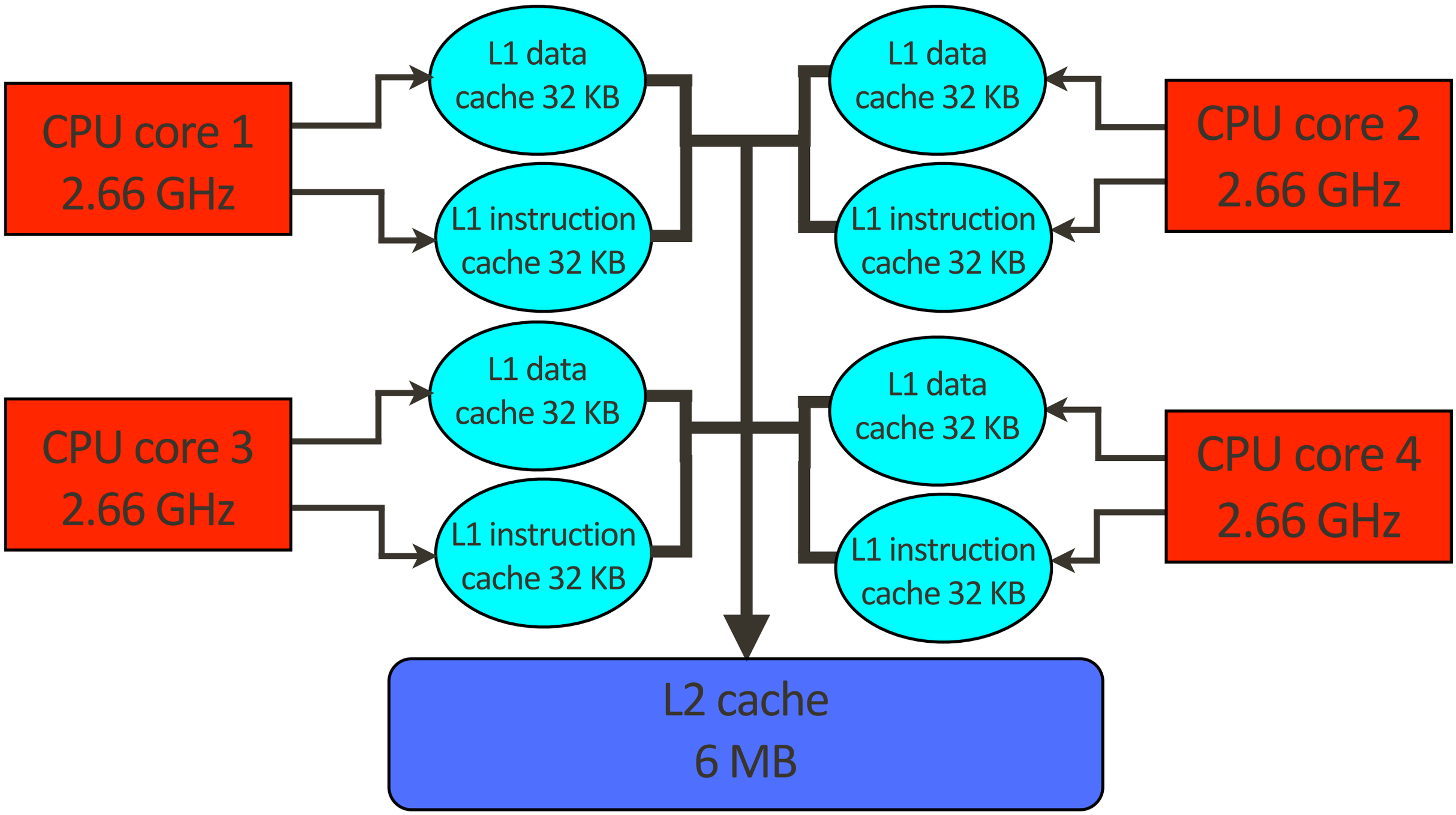
LIMA cluster node

- Intel Xeon E5430 quad core, dual CPU
- Each CPU has its own Front Side Bus for communication with memory



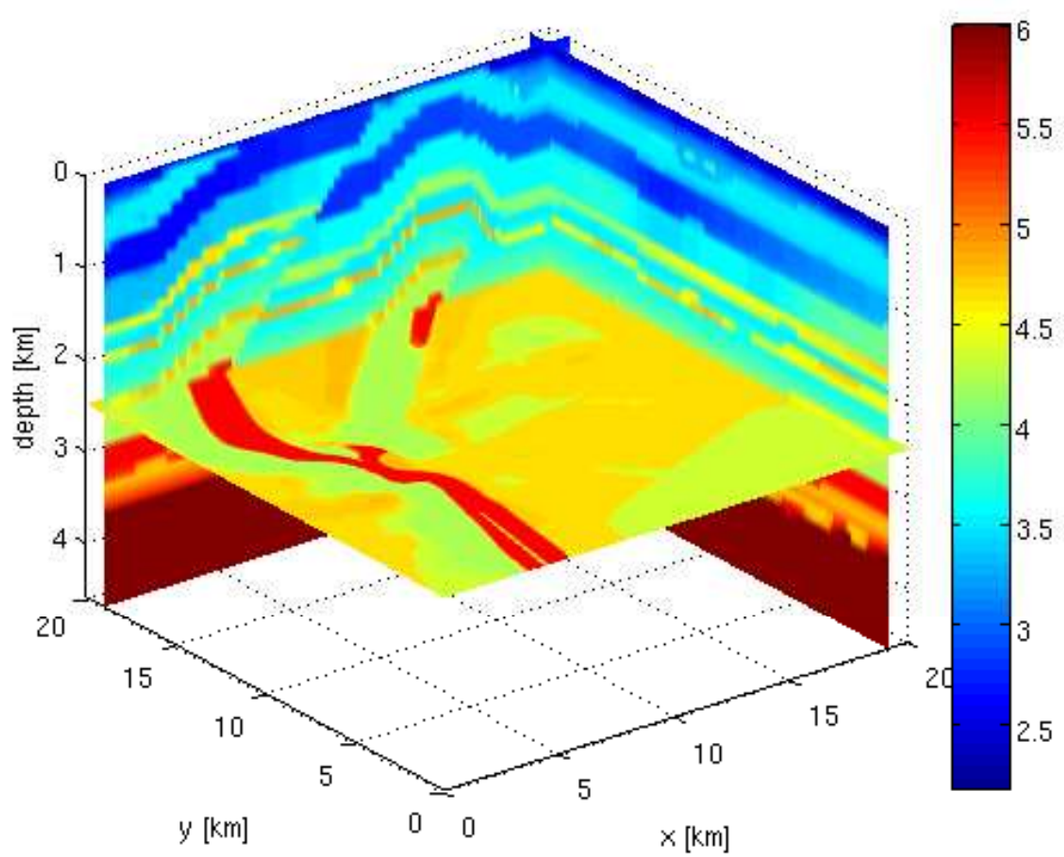
source: http://ark.intel.com/products/33081/Intel-Xeon-Processor-E5430-12M-Cache-2_66-GHz-1333-MHz-FSB

Intel Xeon E5430 layout

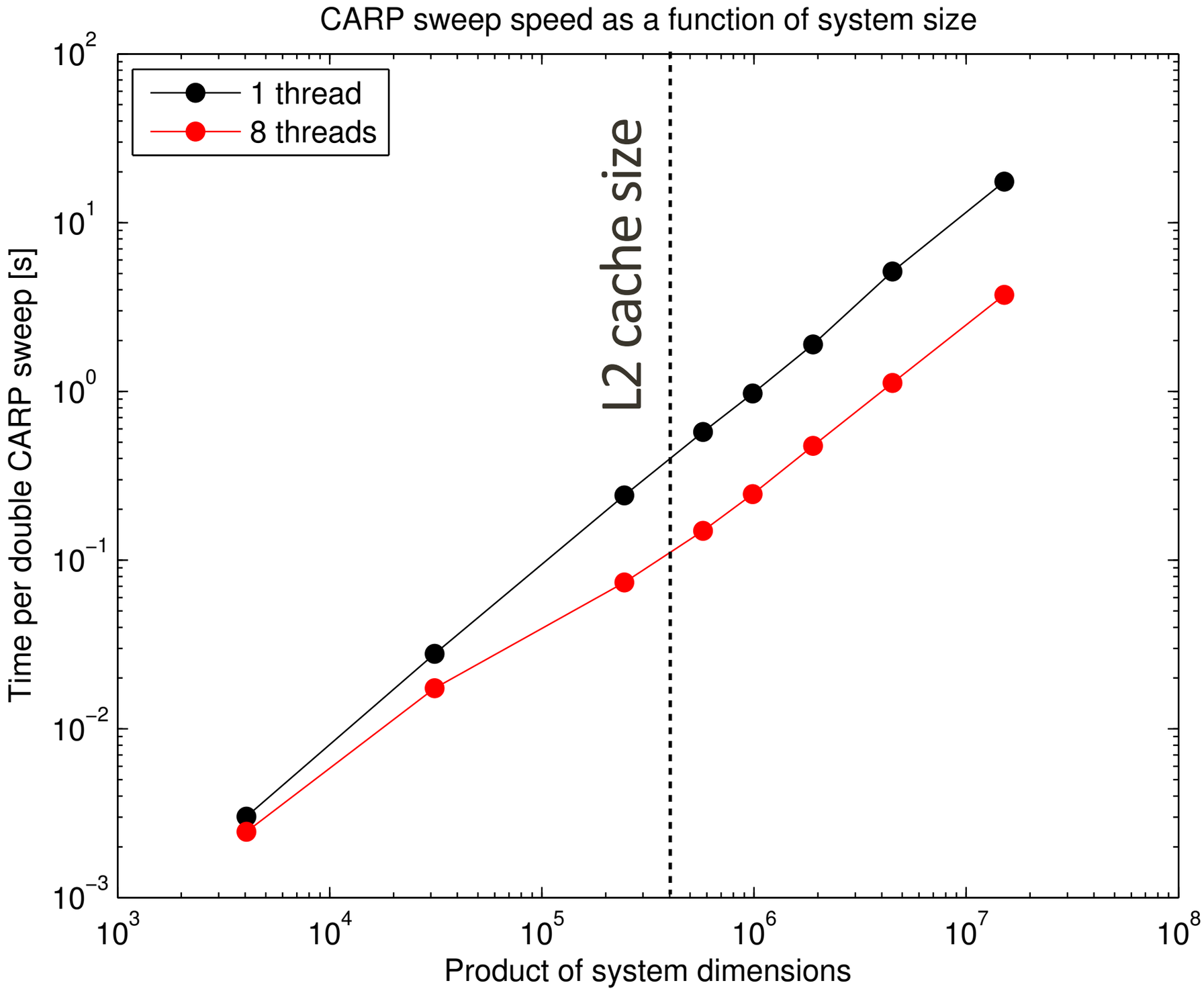


CARP Experiment set-up

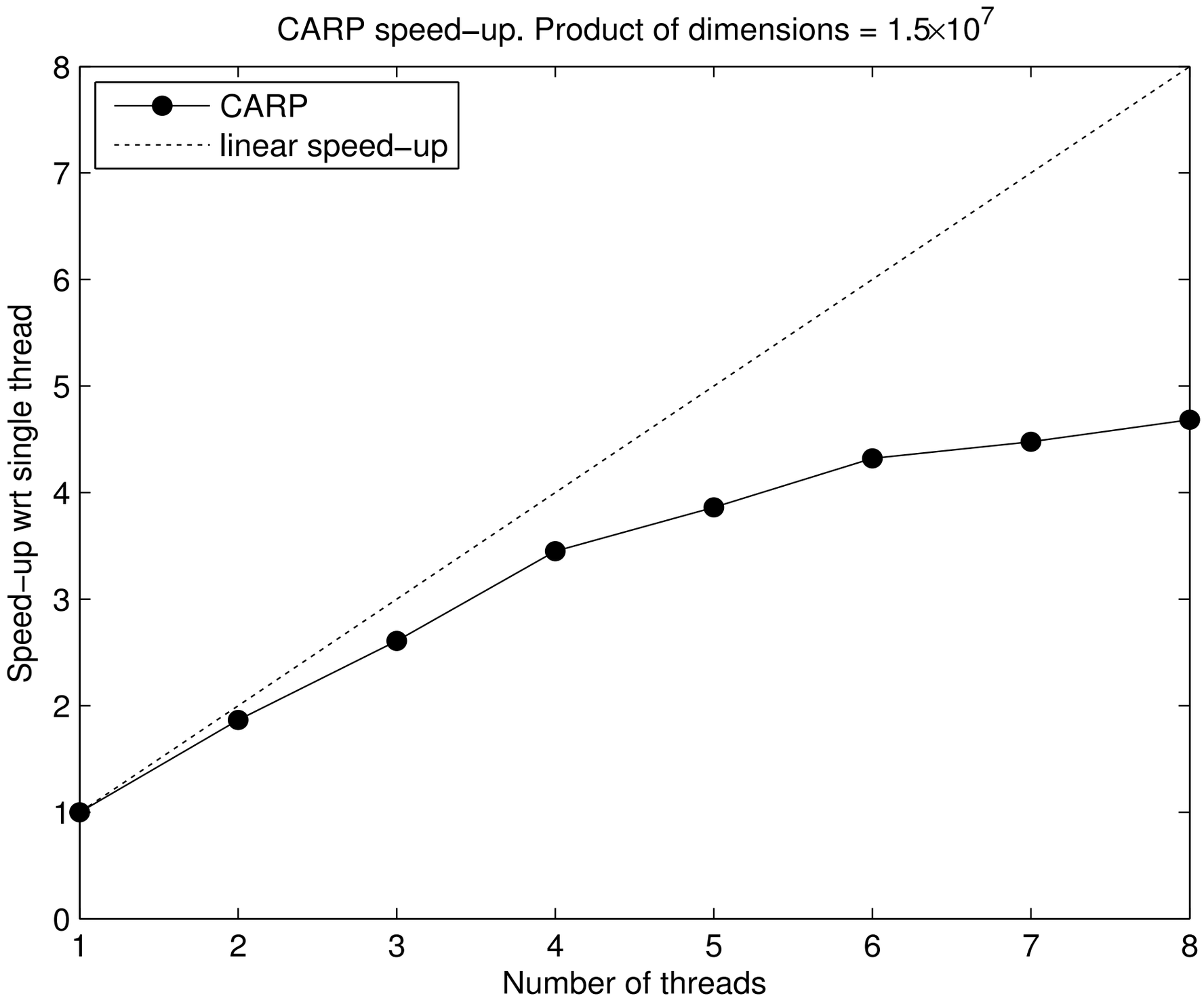
- SEG/EAGE Overthrust model, under-sampled with factors of 2—6, 8, 16, 32.
- Simultaneous source throughout the domain, with normally distributed amplitude.
- w = 1.5
- Zero initial guess.
- Timings are per one CARP double sweep.
- Timings of five runs are averaged for each combination of (system size, degree of parallelism)
- Frequency = 12 Hz / under-sampling



Timing CARP


CARP sweep speed as a function of system size
1 thread
8 threads
L2 cache size
Time per double CARP sweep [s]
Product of system dimensions
10^{2}
10^{1}
10^{0}
10^{-1}
10^{-2}
10^{-3}
10^{3}
10^{4}
10^{5}
10^{6}
10^{7}
10^{8}

Timing CARP: speed-up

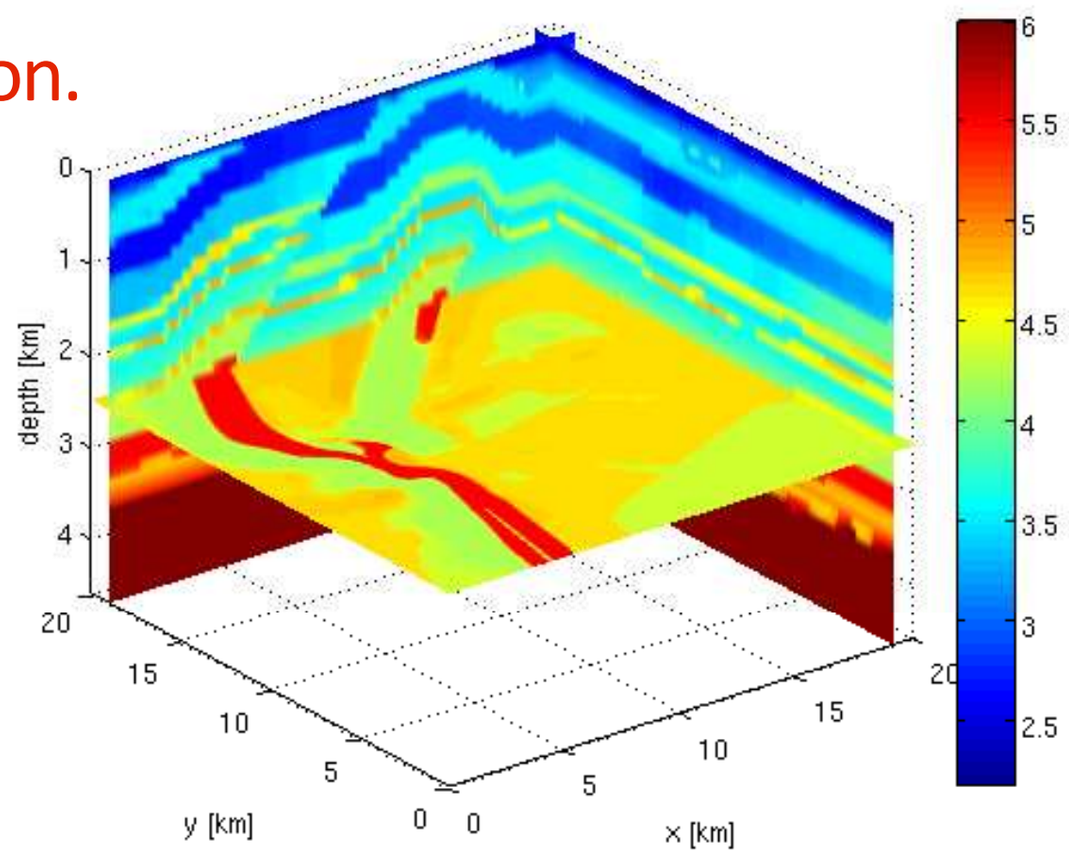


On to CARP-CG: Distributed or shared memory?

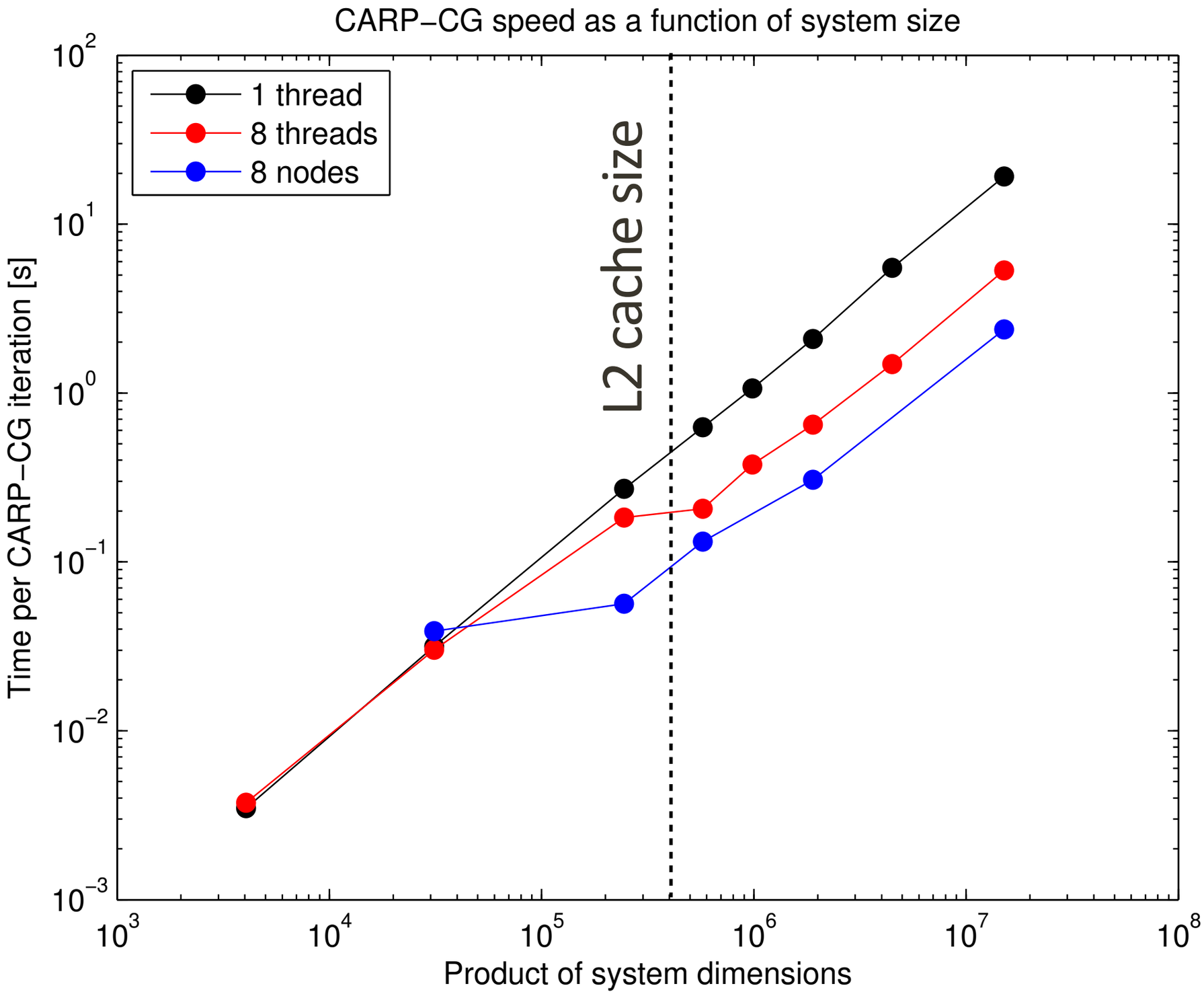
- Distributed setup (parallelization over nodes) has expensive inter-node communication.
- Shared setup (parallelization over threads) has bottle-necks at the cache, memory controller, and memory bus.

CARP-CG Experiment set-up

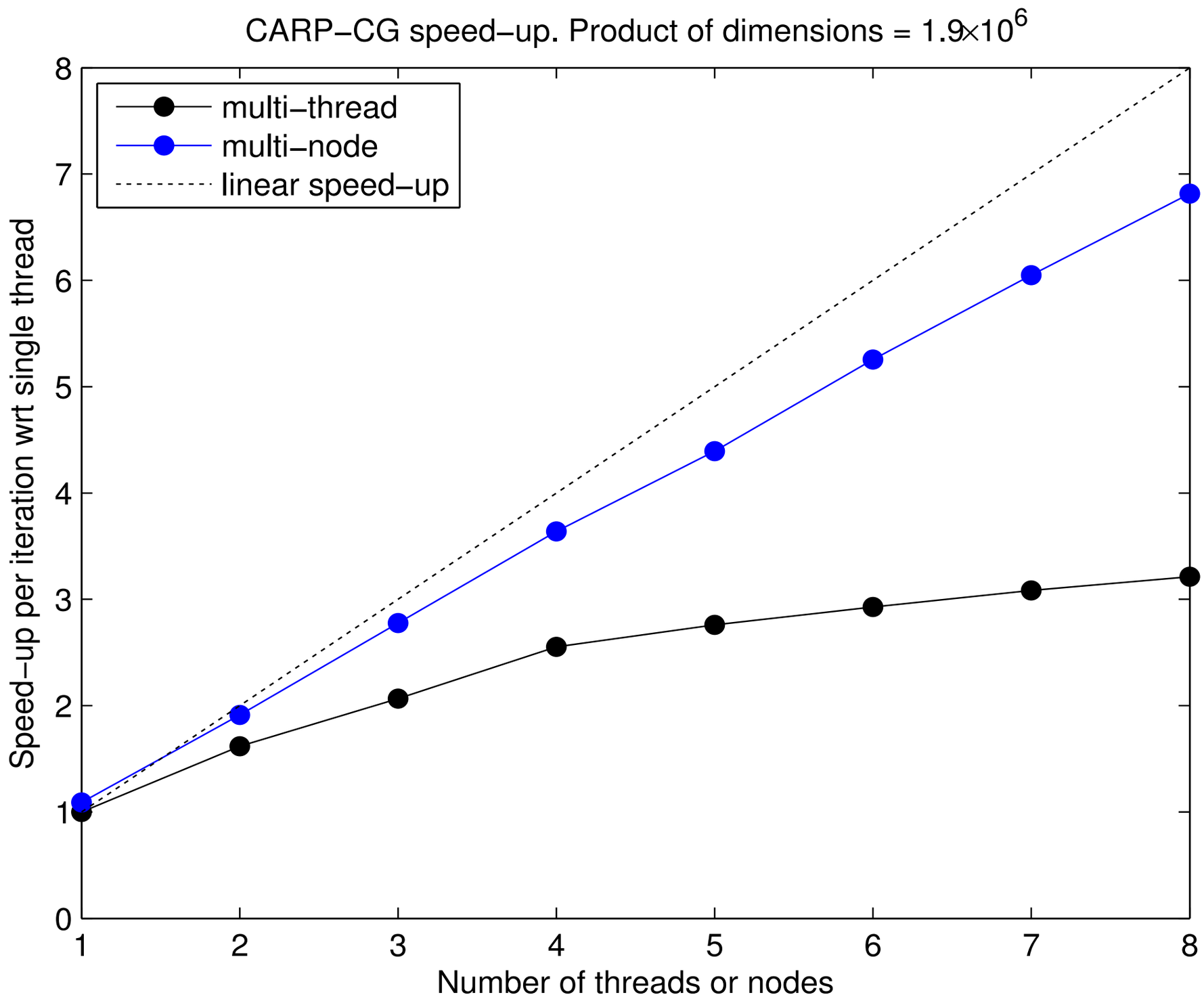
- SEG/EAGE Overthrust model, under-sampled with factors of 2—6, 8, 16, 32.
- Simultaneous source throughout the domain, with normally distributed amplitude.
- w = 1.5
- Zero initial guess.
- Timings are per one CARP-CG iteration.
- CARP-CG converges to a tolerance of 10^{-4}.
- Timings of five runs are averaged for each combination of (system size, degree of parallelism)
- Frequency = 12 Hz / under-sampling



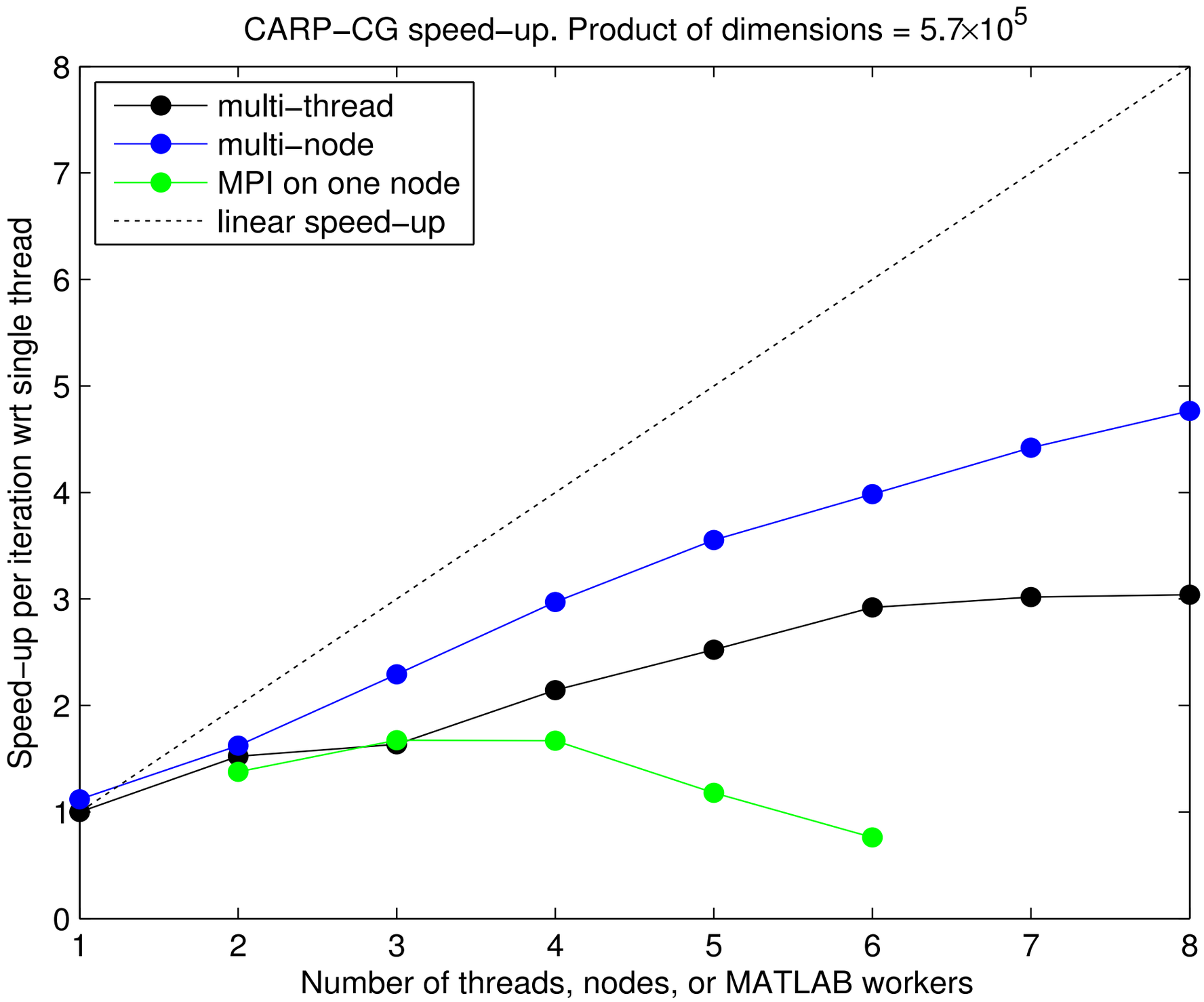
Nodes vs. Threads



Nodes vs. Threads



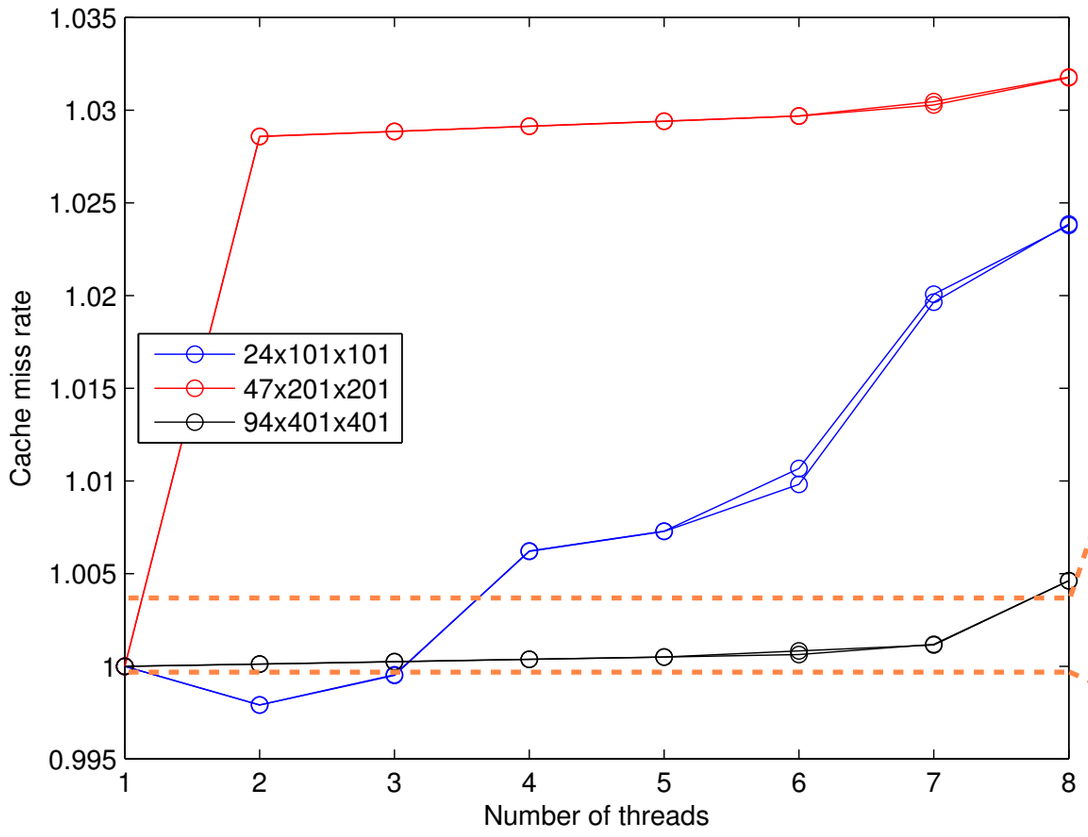
Nodes vs. Threads



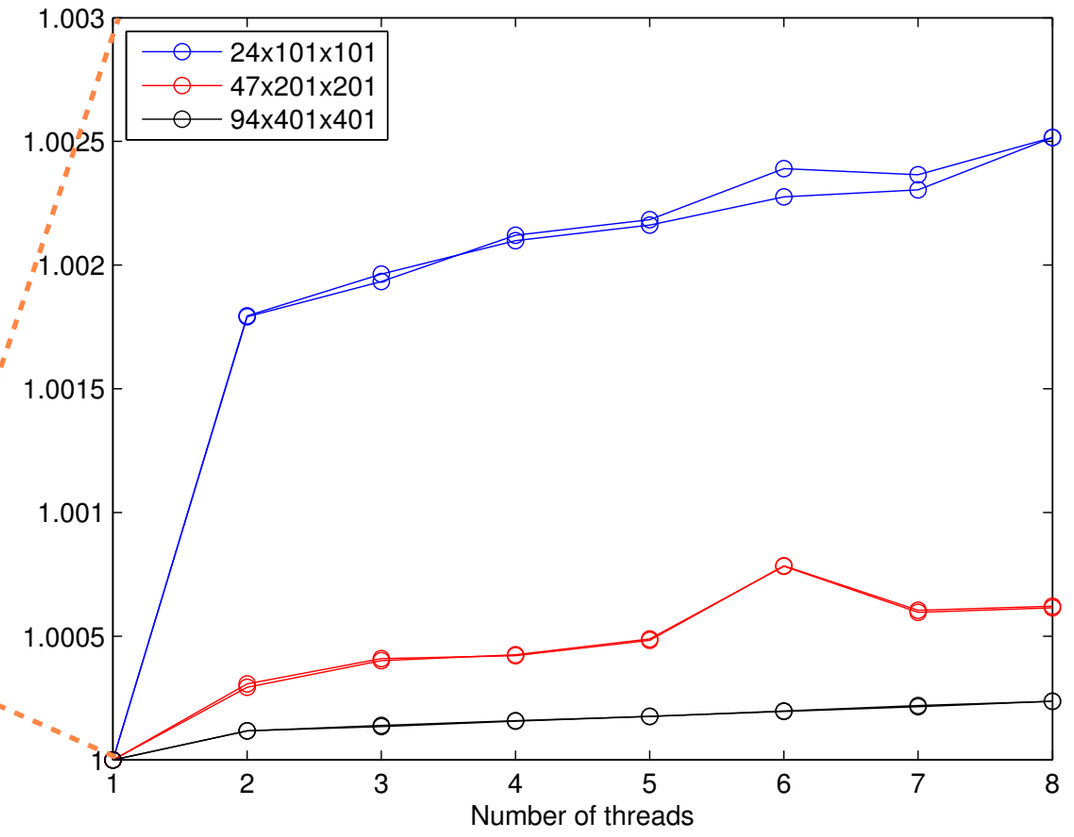
Measuring cache misses

- One forward CARP sweep
- SEG/EAGE overthrust model
- zero initial guess, normally distributed source
- valgrind --tool=cachegrind --fair-sched=yes

L2 cache misses



L1 cache misses



Conclusions and obvious improvements

- Cache misses are not the main reason for slow performance of threaded CARP sweeps.
- Combine MPI across several nodes with multi-threading on each node.
- All of CARP-CG will be re-written in C/MEX (complete), threaded (in progress) and MPI implemented (to be done).
- Intel compiler will be used for finer tuning of the sweeps to the E5430 Xeon chip.

The Future: CARP-CG on Maxeler's Dataflow Engine

- Dataflow models computing as operations on a stream.
- Rather than executing instructions in ***time***, they are laid out in ***space*** on an FPGA.
- After the initial latency, output is one element per clock cycle.
- Even with the large (68 GB/s) memory bandwidth, the problem will still be memory bound.
- To partially alleviate the situation, the Helmholtz matrix can be computed on the fly.
- Based on initial estimates, we anticipate a speed-up of the order of 100.

Acknowledgements

- Sincere thanks to Tristan van Leeuwen, Felix Herrmann and Henryk Modzelewski for helpful discussions, and to members of the SLIM group for feedback on an early version of this presentation.

SINBAD

NSERC CRSNG

This work was in part financially supported by the Natural Sciences and Engineering Research Council of Canada Discovery Grant (22R81254) and the Collaborative Research and Development Grant DNOISE II (375142-08). This research was carried out as part of the SINBAD II project with support from the following organizations: BG Group, BGP, BP, Chevron, ConocoPhillips, Petrobras, PGS, Total SA, and WesternGeco.

Selected references

- **Operto et al. 2007**: Stéphane Operto, Jean Virieux, Patrick Amestoy, Jean-Yves L'Excellent, Luc Giraud and Hafedh Ben Hadj Ali. *3D finite-difference frequency-domain modeling of visco-acoustic wave propagation using a massively parallel direct solver: A feasibility study*. Geophysics 72(5).
- **Gordon & Gordon 2005:** *Component-Averaged Row Projections: A Robust, Block-Parallel Scheme for Sparse Linear Systems*. SIAM J. Sci. Computing 27(3).
- **Gordon & Gordon 2010:** *CARP-CG: A robust and efficient parallel solver for linear systems, applied to strongly convection dominated PDEs*. Parallel Computing 36.
- **van Leeuwen, Gordon & Gordon and Herrmann 2012**: *Preconditioning the Helmholtz equation via row-projections.* EAGE Conference.

MATLAB vs. julia