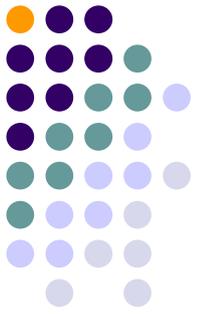


# Future Computer Needs in the Dense Linear Algebra Domain

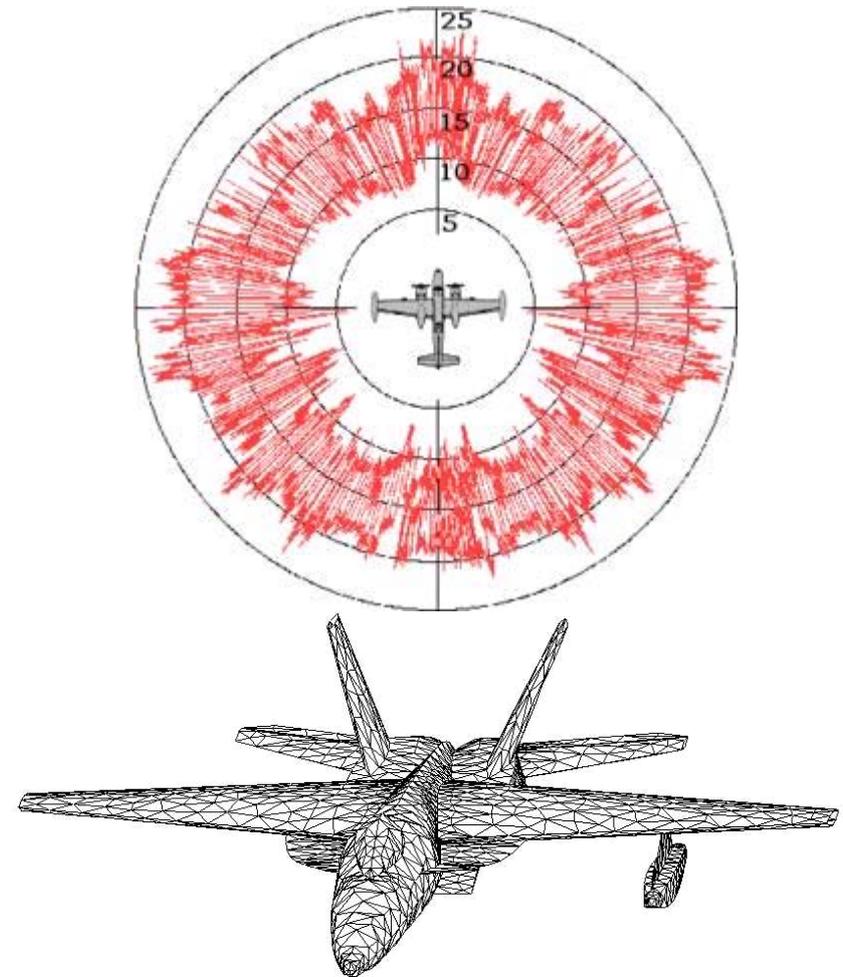
Enrique S. Quintana-Ortí



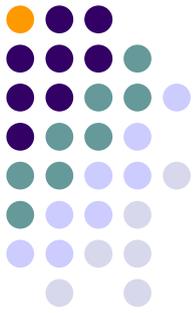
# Large-scale linear systems: Electromagnetism



- Radar cross-section problem (via BEM)
- Solve  $A x = b$   
dense  $A \rightarrow n \times n$   
 $n =$  hundreds of  
thousands of boundary  
points (or panels)



# Large-scale LLS: Estimation of Earth's gravity field



- GRACE project

[www.csr.utexas.edu/grace](http://www.csr.utexas.edu/grace)

- Solve  $y = H x_0 + \epsilon$ ,  
dense  $H \rightarrow m \times n$   
 $m = 66.000$  observations  
 $n = 26.000$  parameters for  
a model of resolution  
250km

**GRACE** Gravity Recovery and Climate Experiment

HOME | SCIENCE | OPERATIONS | MISSION | FLIGHT SYSTEMS | CSR  
GAMES | EDUCATION | PUBLICATIONS | GALLERY | SEARCH

The GGM02 Models  
Science Data Products  
Level-3 Data Products  
GRACE Science Team Meeting  
Mission Overview  
GRACE Launch  
GRACE Newsletter  
GRACE Partners  
PO DAAC  
ISDC  
GRACE Tellus  
Science Citations

GRACE, twin satellites launched in March 2002, are making detailed measurements of Earth's gravity field which will lead to discoveries about gravity and Earth's natural systems. These discoveries could have far-reaching benefits to society and the world's population.

Orbiting Twins - The GRACE satellites

Current Orbit Data

| Mission Elapsed Time |       |
|----------------------|-------|
| Days                 | Hours |
| 3005                 | 04    |

**GRACE in the News**

**GRACE offers broad snapshot of groundwater**

New satellite data showcased at the American Geophysical Union meeting this week in San Francisco illustrated the degree to which groundwater has dropped over the past several years in California's Central Valley.

GRACE, the Gravity Recovery and Climate Experiment, through a partnership with NASA and the German Aerospace Center, tracks the monthly changes in the earth's gravity field caused by the movement of water.

[Learn More](#)

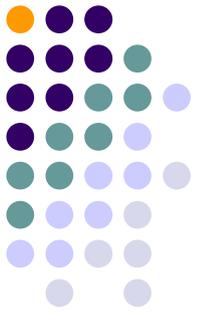
HOME | SCIENCE | OPERATIONS | MISSION | FLIGHT SYSTEMS | CSR |  
GAMES | EDUCATION | PUBLICATIONS | GALLERY | SEARCH |

The GRACE mission is jointly implemented by NASA and DLR under the NASA Earth System Science Pathfinder Program.

Last Modified: Wed Jan 06, 2010  
CSR/TSGC [Team Web](#)

UTOPIA  
THE UNIVERSITY OF TEXAS AT AUSTIN

# Large-scale eigenvalue problems: Industrial processes



- Optimal cooling of steel profiles

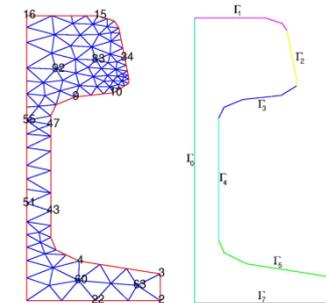


- Solve

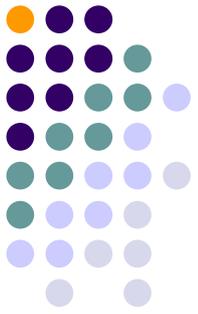
$$A^T X + X A - X S X + Q = 0,$$

dense  $A \rightarrow n \times n$

$n = 5.177$  for a mesh  
width of  $6.91 \cdot 10^{-3}$

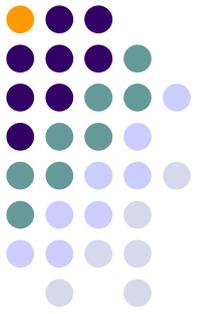


# Other applications



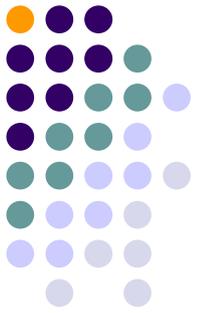
- Dense linear algebra is at the bottom of the “food chain” for many scientific and engineering apps.
- Molecular dynamics simulations
- Fast acoustic scattering problems
- Dielectric polarization of nanostructures
- Magneto-hydrodynamics
- Macro-economics

# Challenges for dense linear algebra



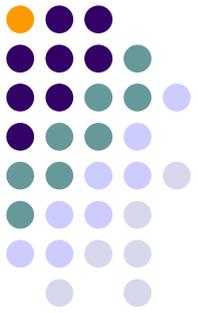
- Future exascale platforms
  - Performance scalability
  - Architecture heterogeneity
  - Power consumption
- Impact on methods and libraries for dense linear algebra operations...

# Challenges for dense linear algebra



- Future exascale platforms
  - Performance scalability
  - Architecture heterogeneity
  - Power consumption
- Impact on methods and libraries for dense linear algebra operations...

# Performance scalability



## PFLOPS ( $10^{15}$ flops/sec.)

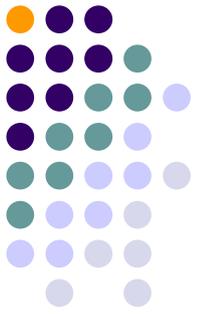
### 2010 JUGENE

- $10^9$  core level  
(3.4 GFLOPS)
- $10^1$  node level  
(Quad-Core)
- $10^5$  cluster level  
(73.728 nodes)

## EFLOPS ( $10^{18}$ flops/sec.)

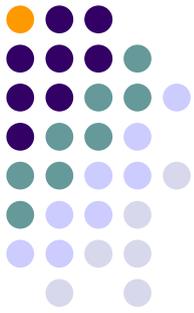
- $10^{9.5}$  core level
- **$10^3$  node level!**
- $10^{5.5}$  cluster level

# Performance scalability

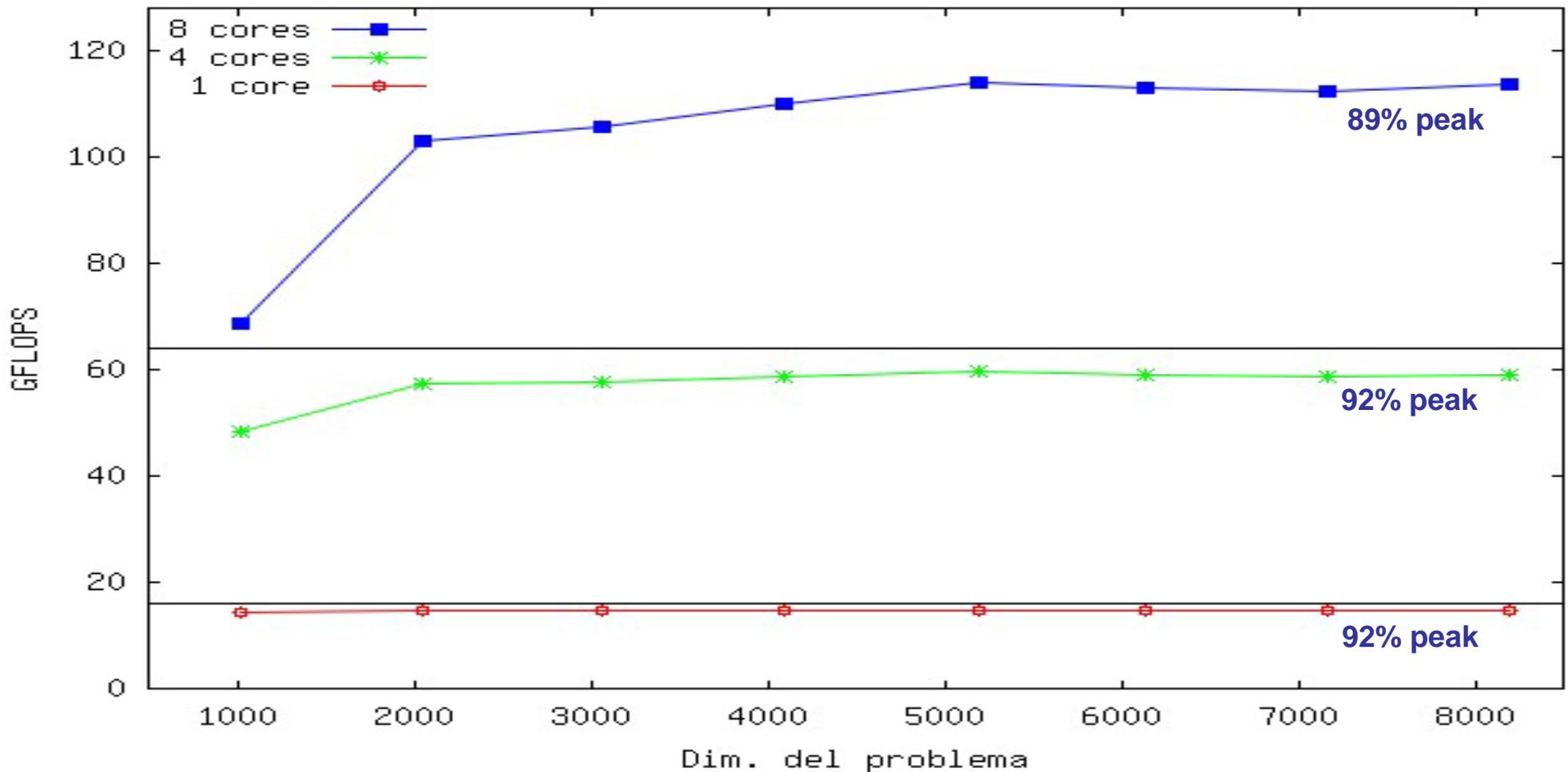


- Libraries for dense linear algebra:
  - BLAS
    - Multi-threaded (MT)
  - LAPACK
    - Use of MT BLAS: Excessive synchronization points
  - ScaLAPACK, PLAPACK:
    - 1 MPI process per core is not optimal: combine with MT-parallel approach

# Performance scalability

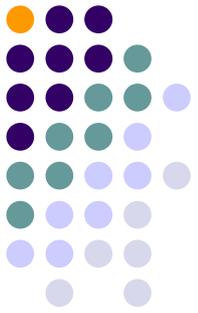


Producto de matrices en 2 Intel Xeon QuadCore (8 cores)



# Performance scalability

## LAPACK



- Cholesky factorization

$$A = L * L^T$$

Key in the solution of s.p.d. linear systems

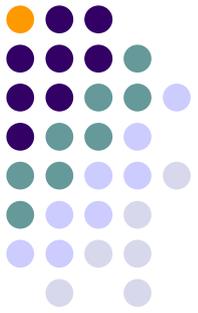
$$A x = b \equiv (L L^T) x = b$$

$$L y = b \Rightarrow y$$

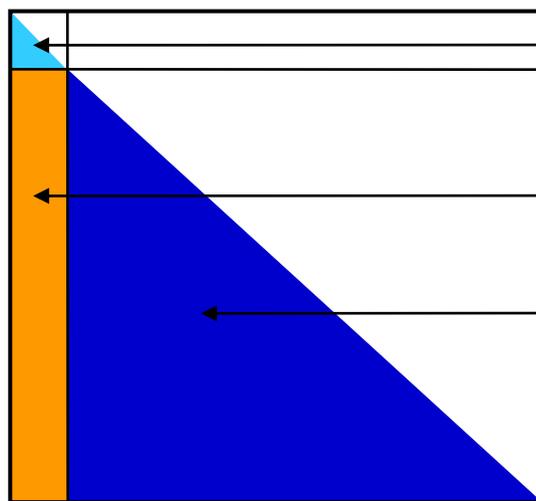
$$L^T x = y \Rightarrow x$$

# Performance scalability

## LAPACK



- Cholesky factorization



1st iteration

**F:**  $A_{11} = L_{11} * L_{11}^T$

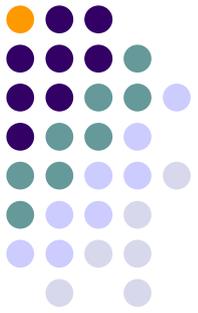
**T:**  $L_{21} \leftarrow A_{21} * L_{11}^{-T}$

**P:**  $A_{22} \leftarrow A_{22} - L_{21} * L_{21}^T$

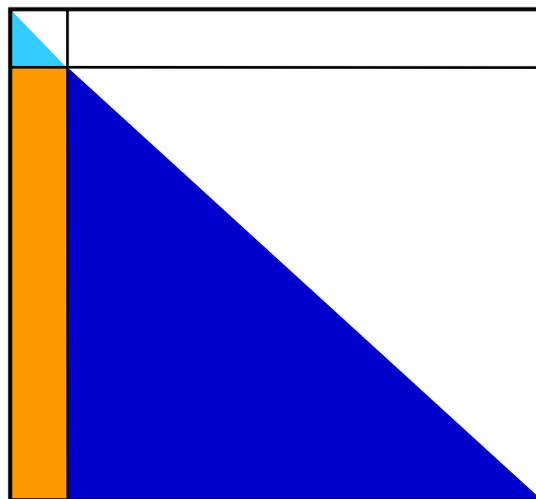
MT processor: Employ a MT implementation of **T** and **P**

# Performance scalability

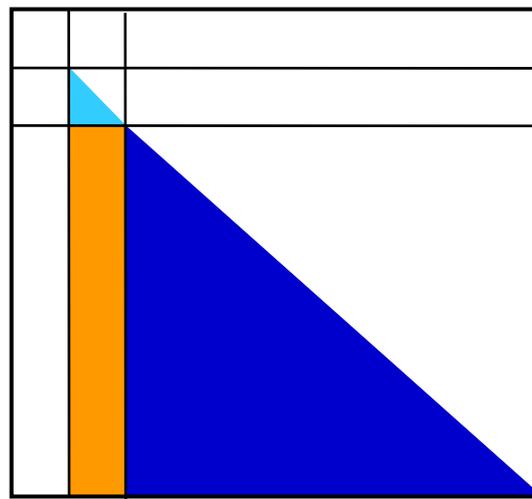
## LAPACK



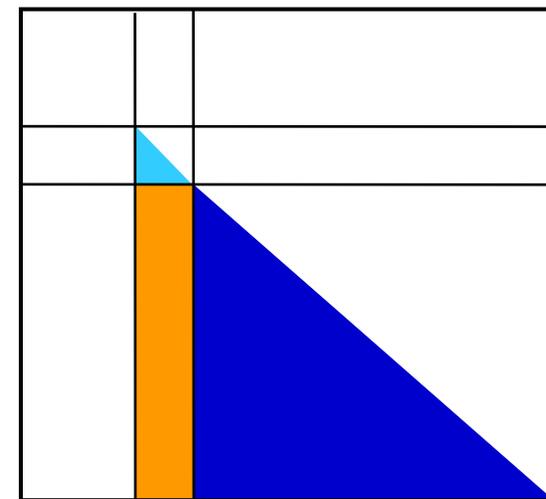
- Cholesky factorization



1st iteration



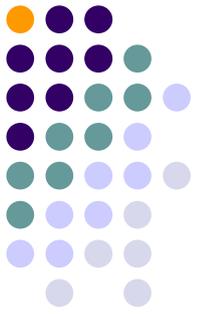
2nd iteration



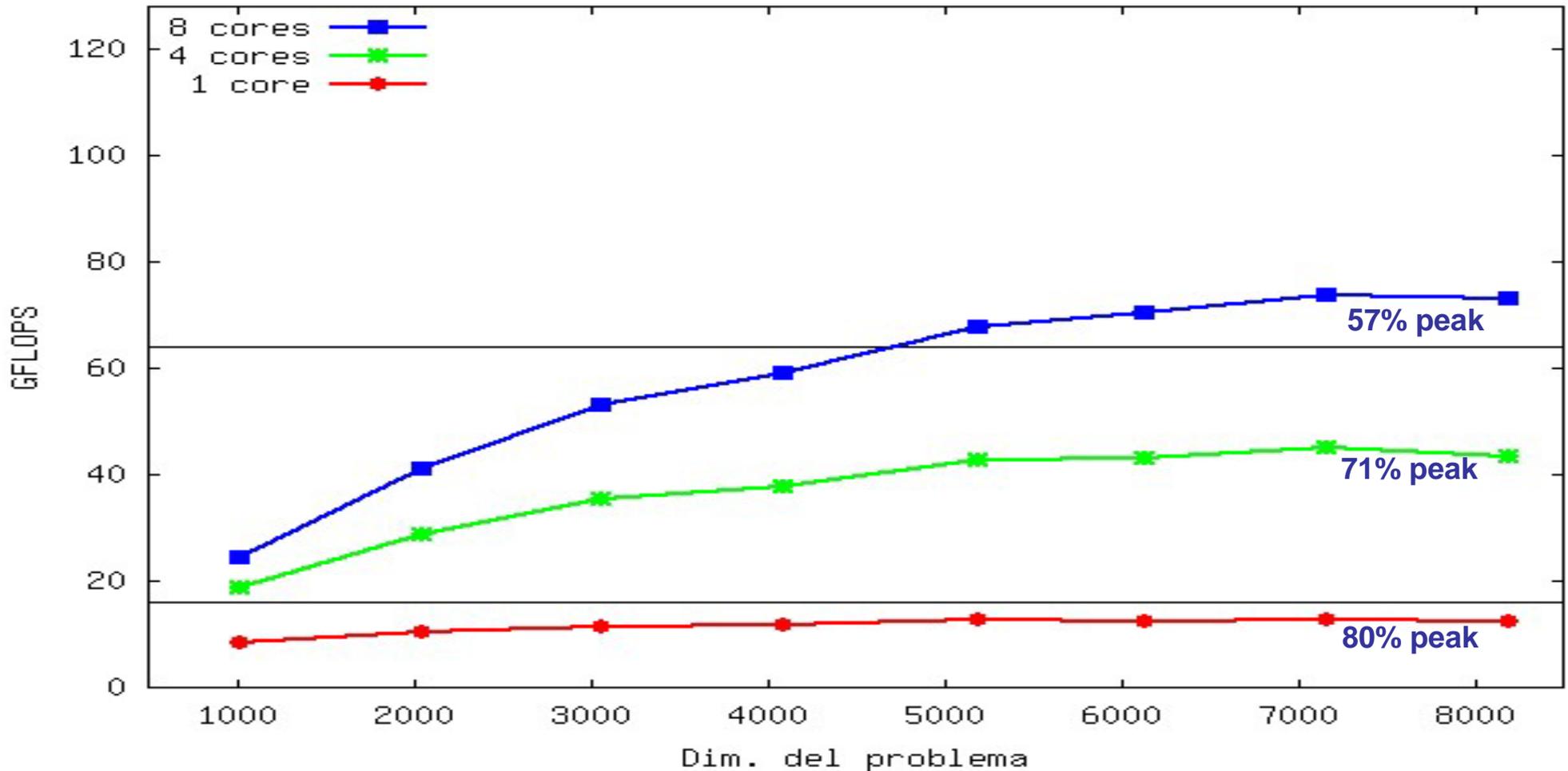
3rd iteration

...

# Performance scalability LAPACK

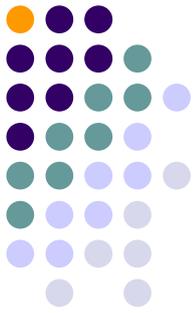


Factor de Cholesky en 2 Intel Xeon QuadCore (8 cores)



# Performance scalability

## LAPACK

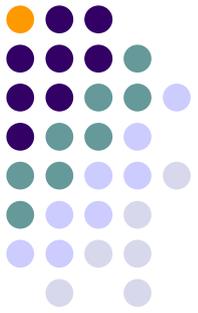


- Why?

### Excessive thread synchronization

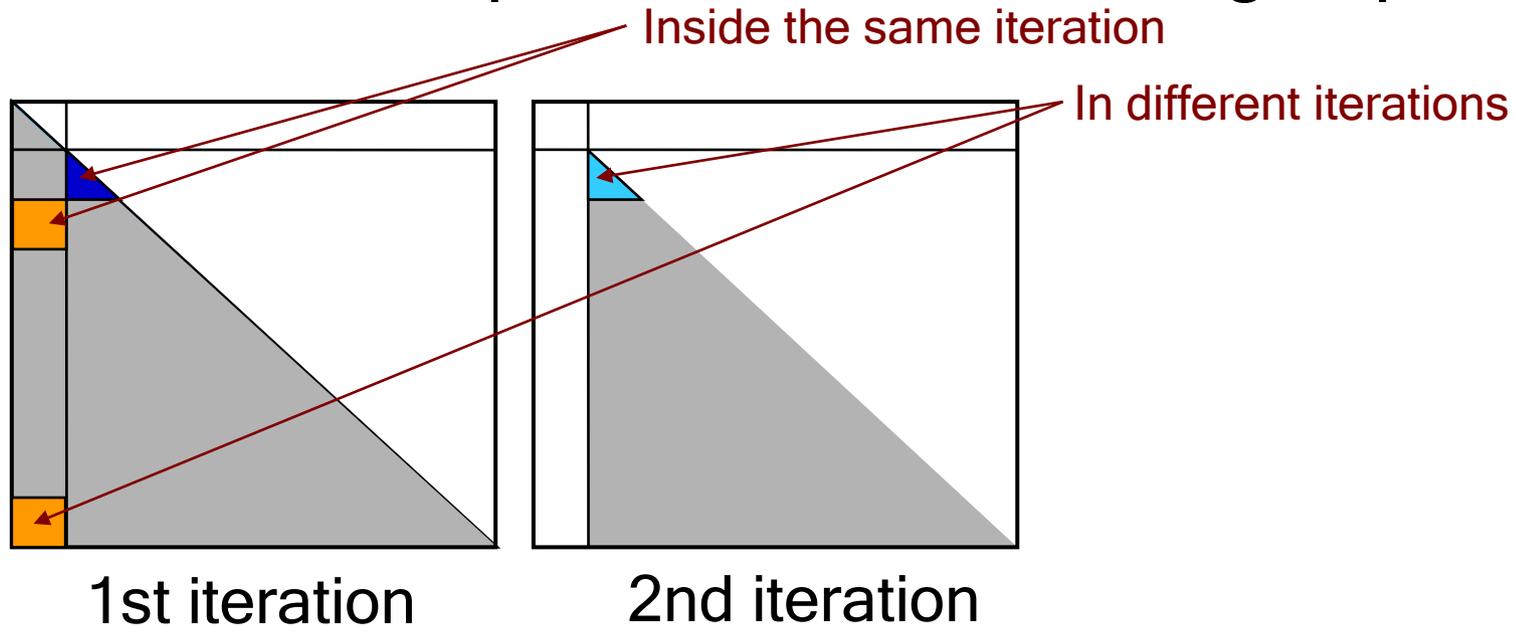
```
for (k=0; k<nb; k++){
    Chol(A[k,k]); //  $A_{kk} = L_{kk} * L_{kk}^T$ 
    if (k<nb){
        Trsm(A[k,k], A[k+1,k]); //  $L_{k+1,k} = A_{k+1,k} * L_{kk}^{-T}$ 
        Syrk(A[k+1,k], A[k+1,k+1]); //  $A_{k+1,k+1} = A_{k+1,k+1}$ 
        //  $- L_{k+1,k} * L_{k+1,k}^T$ 
    }
}
```

# Performance scalability LAPACK



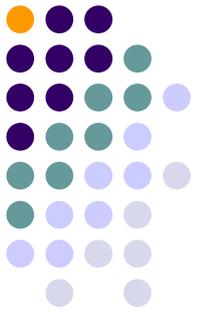
- Why?

There is more parallelism than being exploited



# Performance scalability

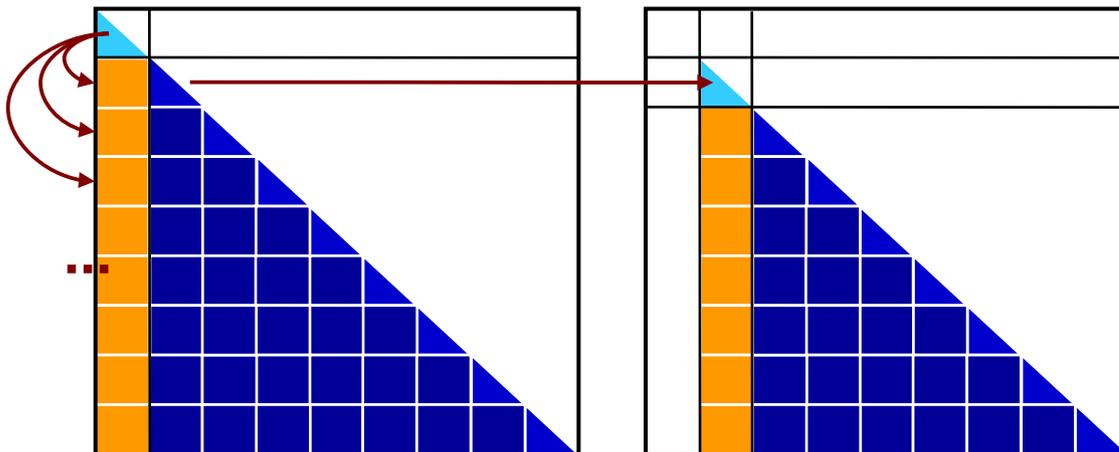
## LAPACK



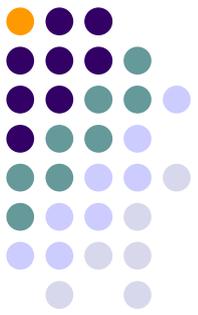
- Exploit task-level parallelism dictated by data dep.

```
for (k=0; k<nb; k++){  
    Chol(A[k,k]);  
    for (i=k+1; i<nb; i++){  
        Trsm(A[k,k], A[i,k]); ...  
    }  
}
```

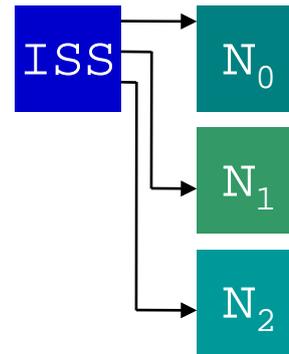
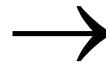
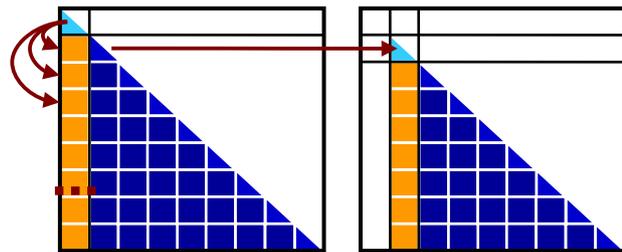
Dependencies among tasks define a tree



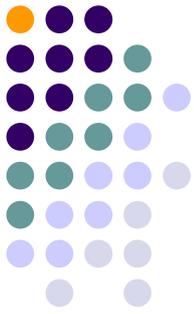
# Performance scalability LAPACK



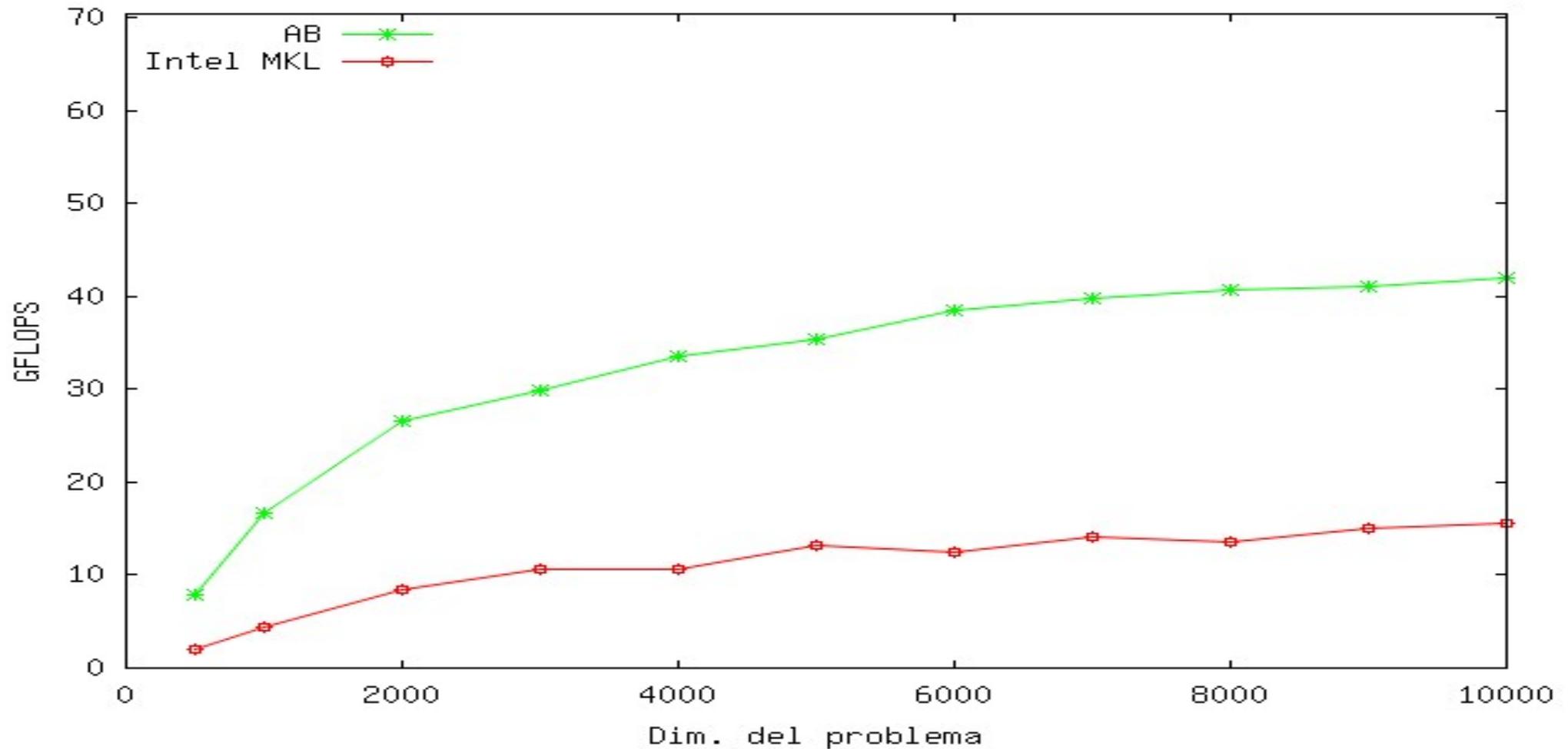
- Run-time:
  - Identifies/extracts TLP
  - Schedules tasks to execution
  - Maps tasks onto specific cores



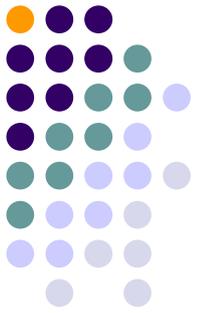
# Performance scalability LAPACK



Factor de Cholesky en 8 AMD Opteron DualCore (16 cores)

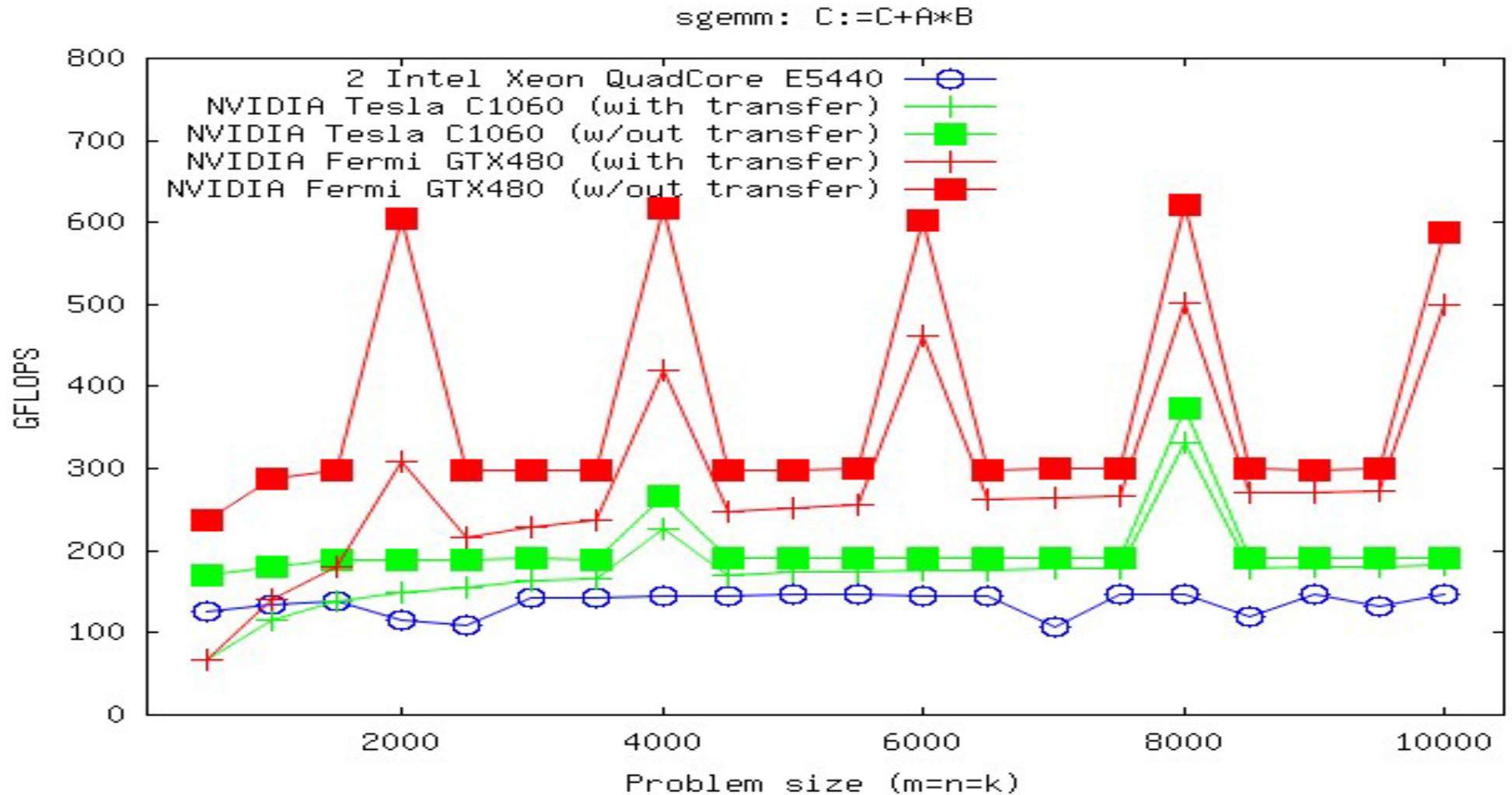
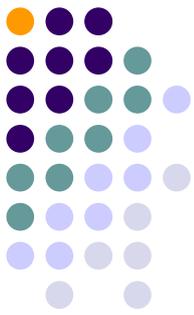


# Challenges for dense linear algebra

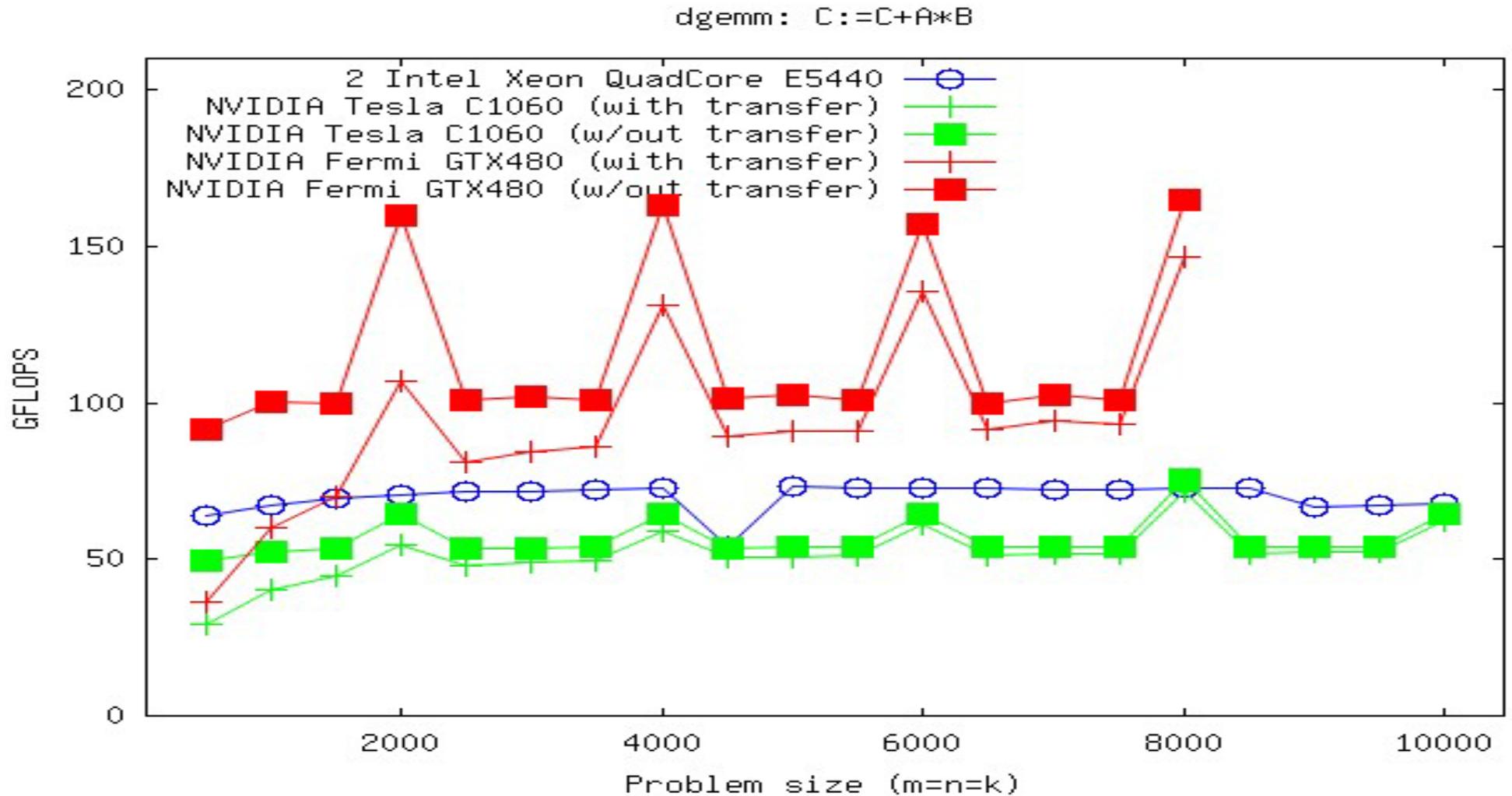
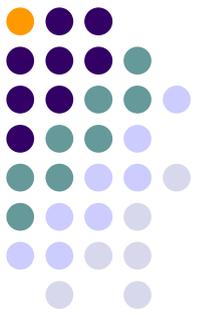


- Future exascale platforms
  - Performance scalability
  - Architecture heterogeneity
  - Power consumption
- Impact on methods and libraries for dense linear algebra operations...

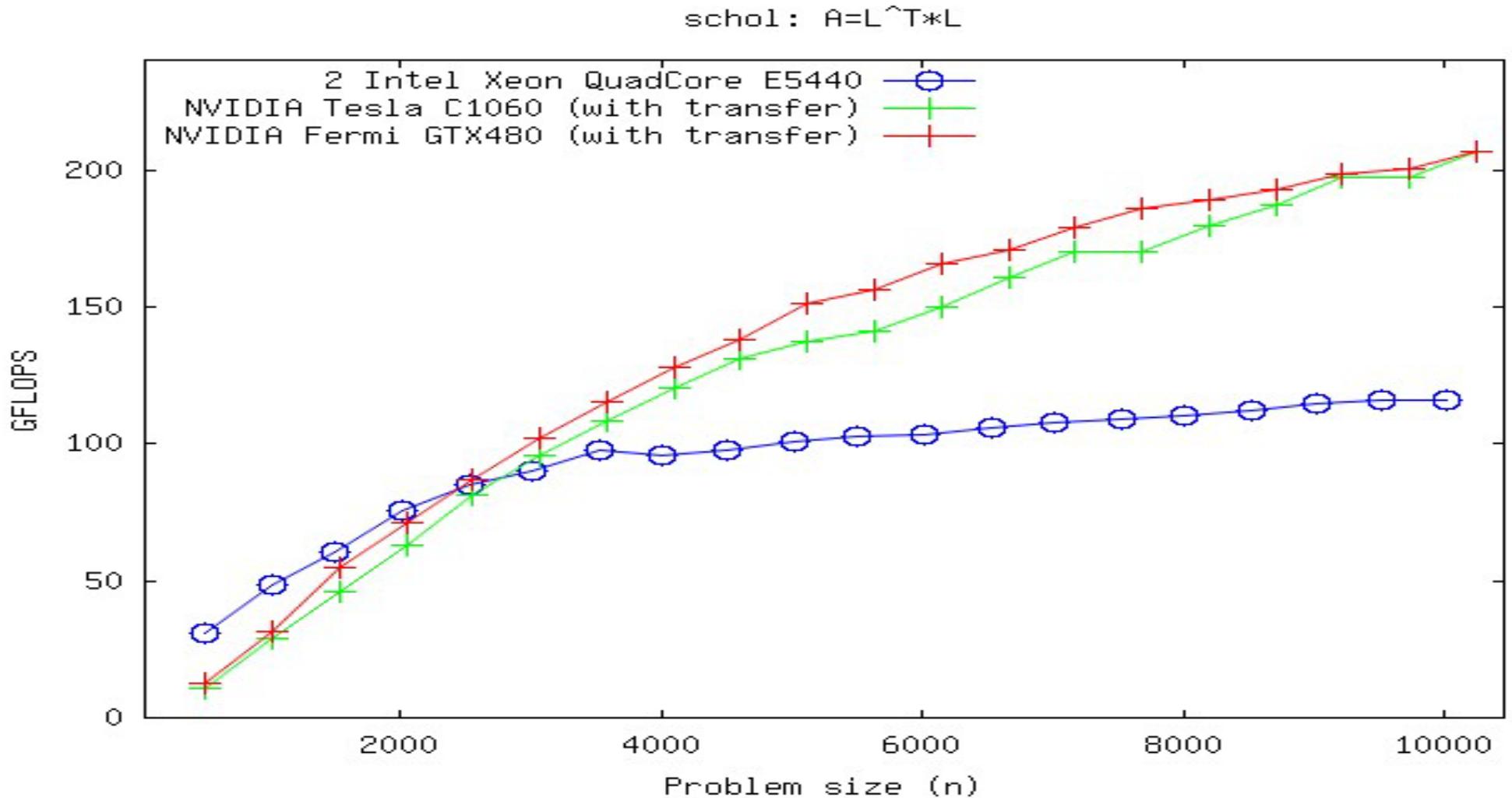
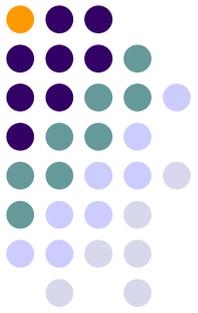
# Architecture heterogeneity (CU)BLAS



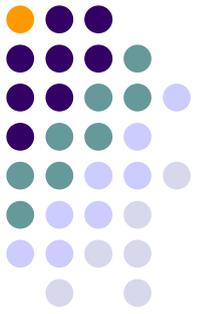
# Architecture heterogeneity (CU)BLAS



# Architecture heterogeneity (CU)LAPACK

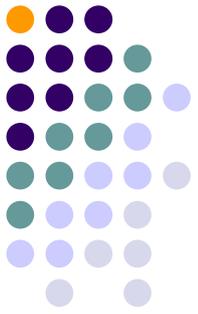


# Architecture heterogeneity

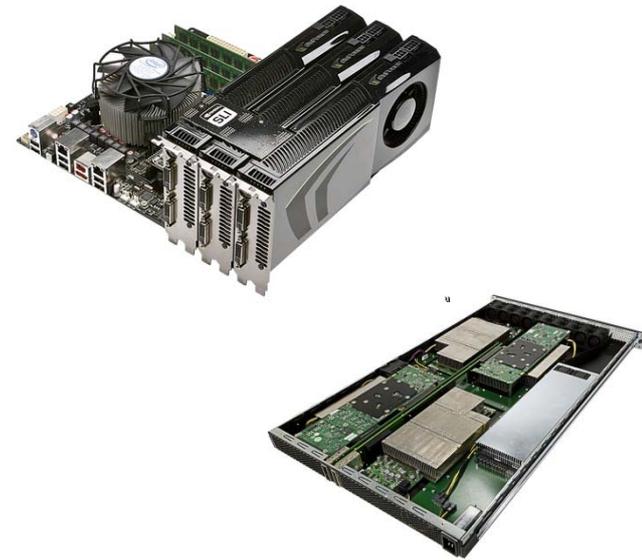
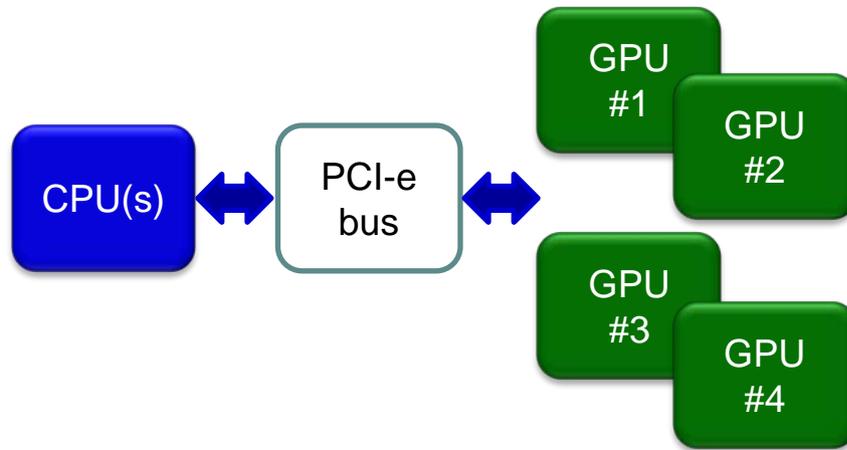


- Libraries for dense linear algebra:
  - Multiple address spaces without hardware coherence (as difficult as message-passing)
  - Scheduling on heterogeneous resources (also much harder)
  - Possibly, more than one accelerator (per node)
  - Take advantage of single precision speed-up: iterative refinement

# Architecture heterogeneity



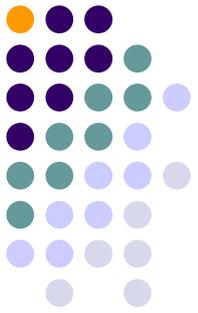
- How do we program these?



View as a...

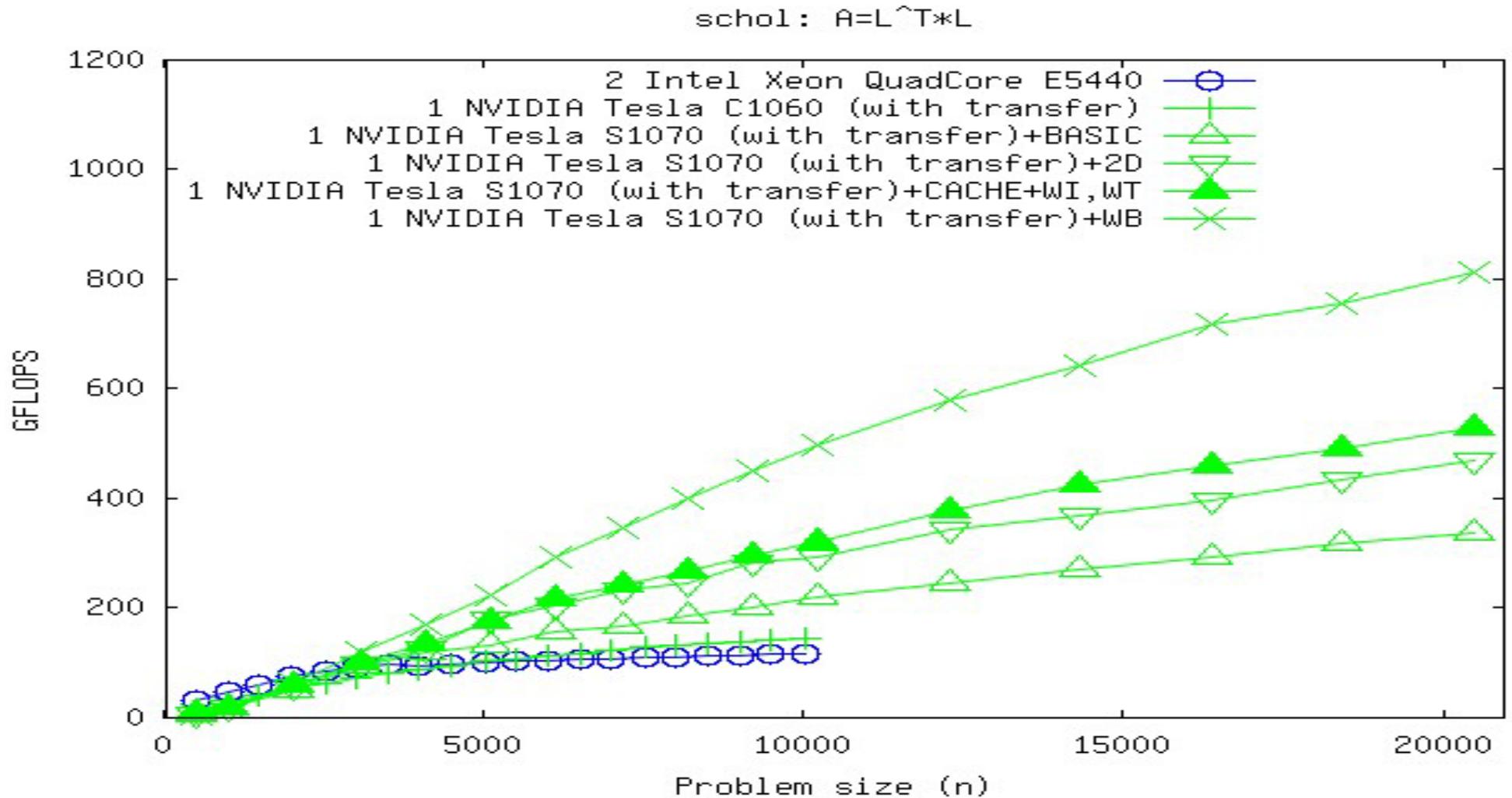
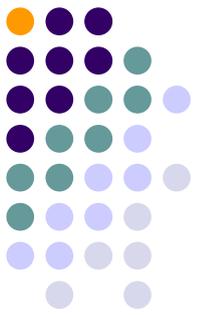
- Shared-memory multiprocessor + DSM

# Architecture heterogeneity

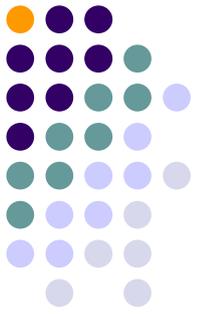


- Software Distributed-Shared Memory (DSM)
  - Software: flexibility vs. efficiency
  - Underlying distributed memory hidden from the users
  - Reduce memory transfers using write-back, write-invalidate,...
  - Well-known approach, not too efficient as a middleware for general apps.
  - Regularity of dense linear algebra operations makes a difference!

# Architecture heterogeneity (CU)LAPACK for multi-GPU



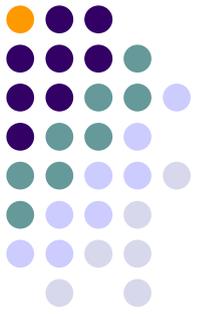
# Challenges for dense linear algebra



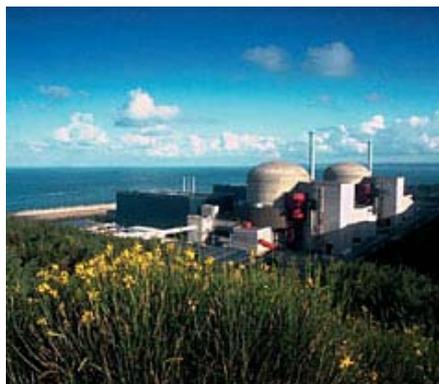
- Future exascale platforms
  - Performance scalability
  - Architecture heterogeneity
  - Power consumption
- Impact on methods and libraries for dense linear algebra operations...

# Power consumption

## Fuel efficiency

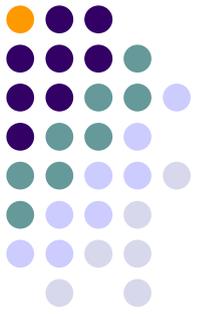


| System       | Top500 | #cores  | Rmax (TFLOPS) | Green500 | Power (KW) | MFLOPS/W | W to EFLOPS? (MW) |
|--------------|--------|---------|---------------|----------|------------|----------|-------------------|
| Jaguar       | 1      | 224,162 | 1,759.0       | 56       | 6,950.6    | 253.1    | 3,951             |
| JUGENE       | 5      | 294,912 | 825.5         | 19       | 2,268.0    | 364.0    | 2,747             |
| FZJ<br>QPACE | 131    | 4,608   | 44.5          | 1        | 253.1      | 773.0    | 1,293             |



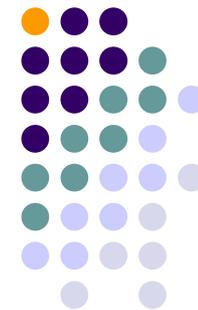
Most powerful reactor under construction in France  
Flamanville: 1,630 MWe

# Power consumption

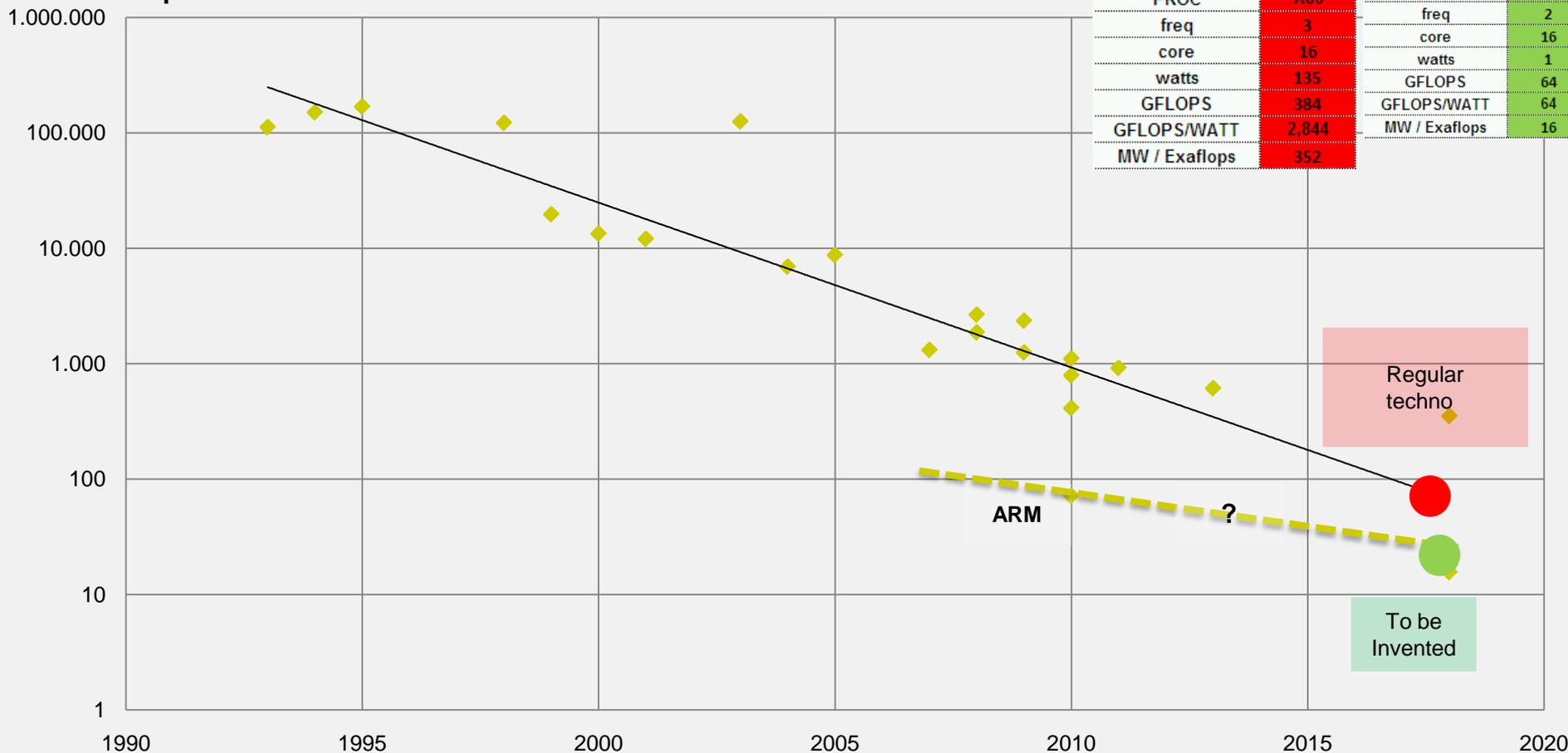


- Future exascale platforms
  - Current approach of “few” (10.000-100.000) and “thick” nodes may not scale
  - “Many thin” nodes (MPI parallelism)?

# Power consumption

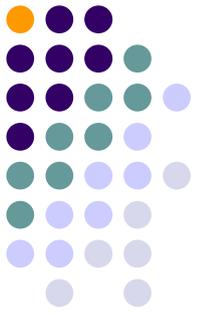


MW / Exaflops



# Power consumption

## Dense linear algebra

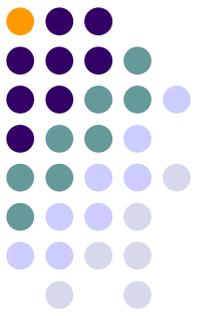


- Need for energy-aware algorithms...
- Example\*: solution of dense s.p.d. linear systems
  - Conjugate gradient (single precision) + iterative refinement
  - Cholesky factorization

\*From “A new energy aware performance metric”; C. Bekas, A. Curioni;  
Comput. Sci. Res. Dev., 2010

# Power consumption

## Dense linear algebra

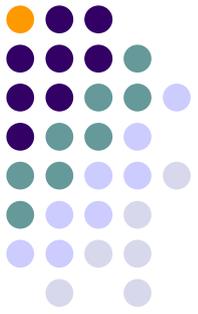


- Execution time in sec. (and percentage of peak performance) for a system with 32,768 unknowns on IBM BG/P, with 4 threads per node

| Solver   | 32 nodes    | 64 nodes    | 128 nodes   |
|----------|-------------|-------------|-------------|
| CG       | 16.2 (4.6)  | 8.1 (4.5)   | 4.2 (4.1)   |
| Cholesky | 51.1 (48.0) | 31.3 (43.0) | 20.3 (33.1) |

# Power consumption

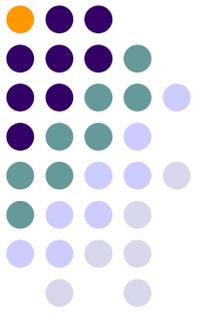
## Dense linear algebra



- Power consumption in KJ for a system with 32,768 unknowns on 128 nodes of IBM BG/P, with 4 threads per node

| Solver   | Flops | Memory | Network | Total |
|----------|-------|--------|---------|-------|
| CG       | 0.11  | 0.12   | 0.0     | 0.23  |
| Cholesky | 3.9   | 8.8    | 0.065   | 12.8  |

# Challenges for dense linear algebra



- Future exascale platforms
  - Performance scalability
  - Architecture heterogeneity
  - Power consumption
- The “battle” will be played at the node level (TLP)
- Heterogeneity is great, but may greatly complicate programming
- Be power efficient or die...