THE INTEL EXASCIENCE PROJECT

HIGH-PERFORMANCE COMPUTING IN THE NEXT DECADE

ROEL WUYTS EXASCIENCE LAB, IMEC

CONTENTS

CARREER OVERVIEW

Studies: Licentiaat Informatica (VUB, 1991-1995)

1995	2001	2004	07	08	09	10	11
Doctoral Researcher VUB	Postdoc University of Bern, Switzerland	Chargé de cours ULB	Principal Scientist imec				
				Professor (10%) KUL			

JUGGLING HATS

ULB
- Object versioning
- AOP

IMEC
- Embedded devices
- Runtime resource management
- Intel ExaScience Project

KUL
Language Design

ExaScience Lab
Intel Labs Europe

ABOUT IMEC

Research organization located in Leuven

- world-leading independent research center in nanoelectronics and nanotechnology
- More Moore research targets semiconductor scaling for the 22nm technology node and beyond.
- More than Moore research invents technology for nomadic embedded systems, wireless autonomous transducer solutions, biomedical electronics, photovoltaics, organic electronics and GaN power electronics.

Numbers

- Budget: ± 280 M€
- Staff: ± 1800
- Cleanroom: ± 10,000 m

EXASCIENCE LAB: NOVEL RESEARCH STRUCTURE

IWT

intel

ExaScience Lab
Intel Labs Europe

Katholieke Universiteit Leuven
Universiteit Gent
Vrije Universiteit Brussel
Universiteit Antwerpen
universiteit hasselt – universiteit van de toekomst

Vlaamse overheid
imec
intel

TOWARDS EXASCALE SUPERCOMPUTERS

SuperComputer ≈ 2 PetaFlops ≈ 100.000 PC's @ 20GigaFlops/PC

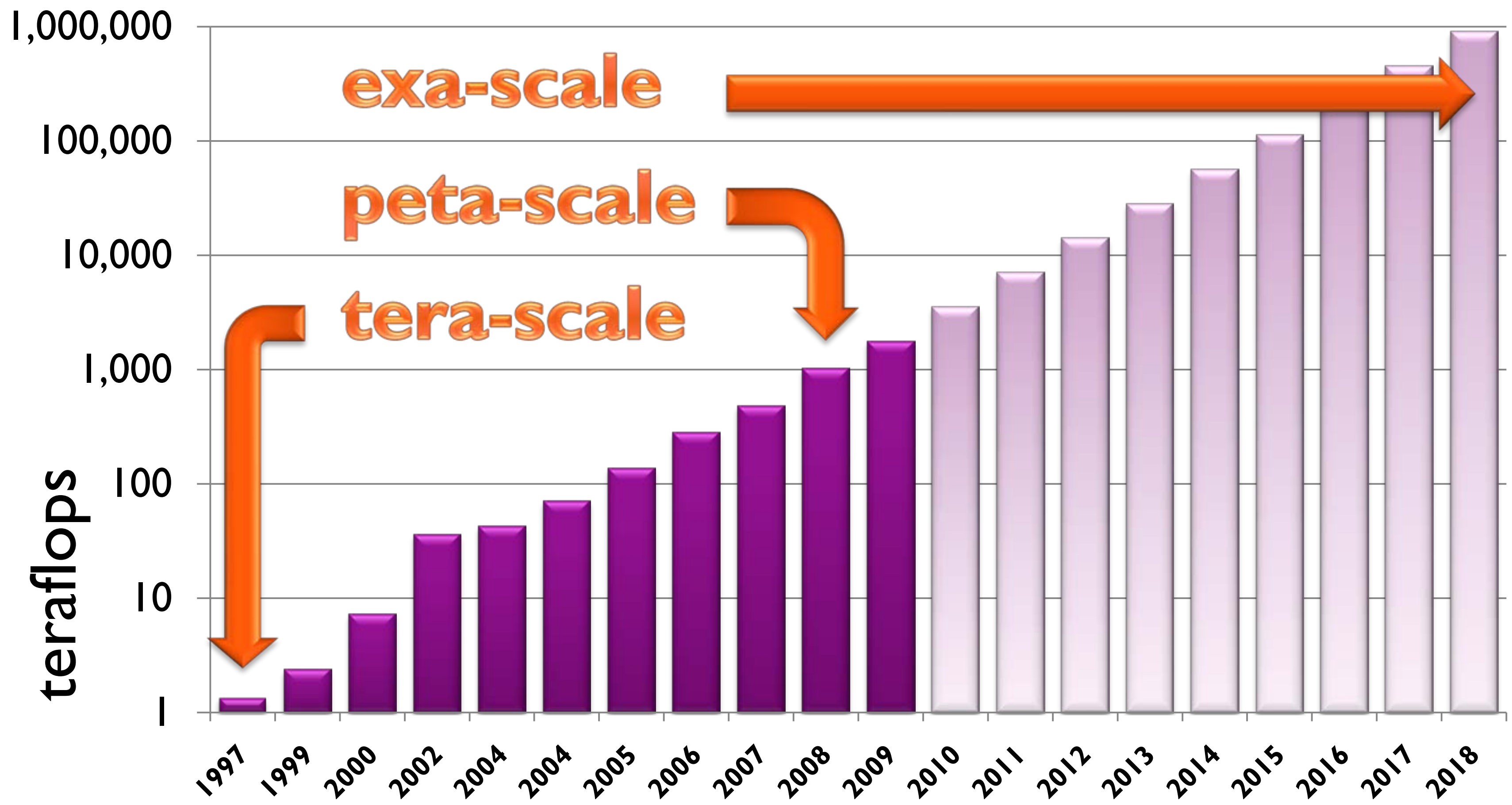
ExaScale ≈ 1000 * PetaScale ≈ 50.000.000 PC's of today

1997	Intel ASCI Red/9152	1
1999	Intel ASCI Red/9632	2
2000	IBM ASCI White	7
2002	NEC Earth Simulator	36
2004	SGI Project Columbia	43
2004	IBM Blue Gene/L	71
2005		137
2006		281
2007		478
2008	IBM Roadrunner	1.026
2009	Cray XT4/XT5 Jaguar	1.759
2010	Tianhe-I	2.5

Kilo	10^3	1.000
Mega	10^6	1.000.000
Giga	10^9	1.000.000.000
Tera	10^{12}	1.000.000.000.000
Peta	10^{15}	1.000.000.000.000.000
Exa	10^{18}	1.000.000.000.000.000.000

ExaScience Lab Intel Labs Europe

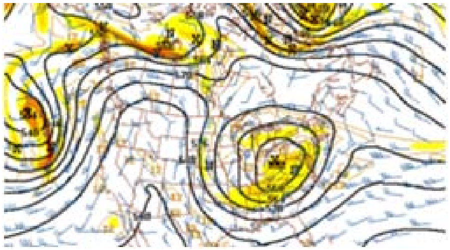
TOWARDS EXASCALE SUPERCOMPUTERS

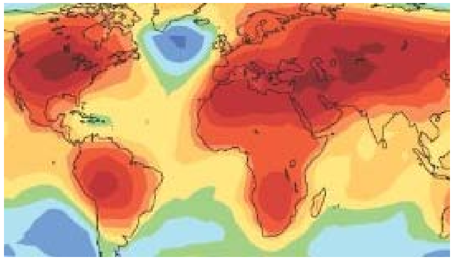


ExaScience Lab
Intel Labs Europe

WHY DO WE NEED THEM ?

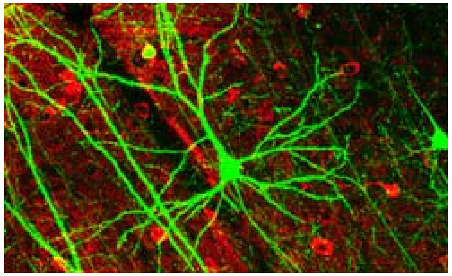
Computer Simulations are needed to fundamentally *understand phenomena* and to accurately *predict.*

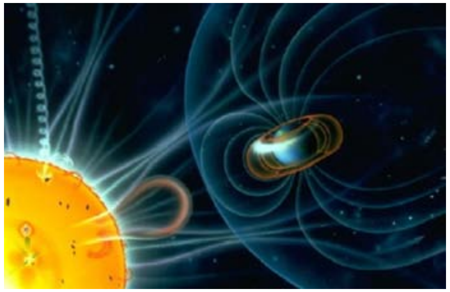




Simulation examples:

- Neural Networks
- Climate Models
- Economical Models
- Weather Prediction
- Space Weather Prediction

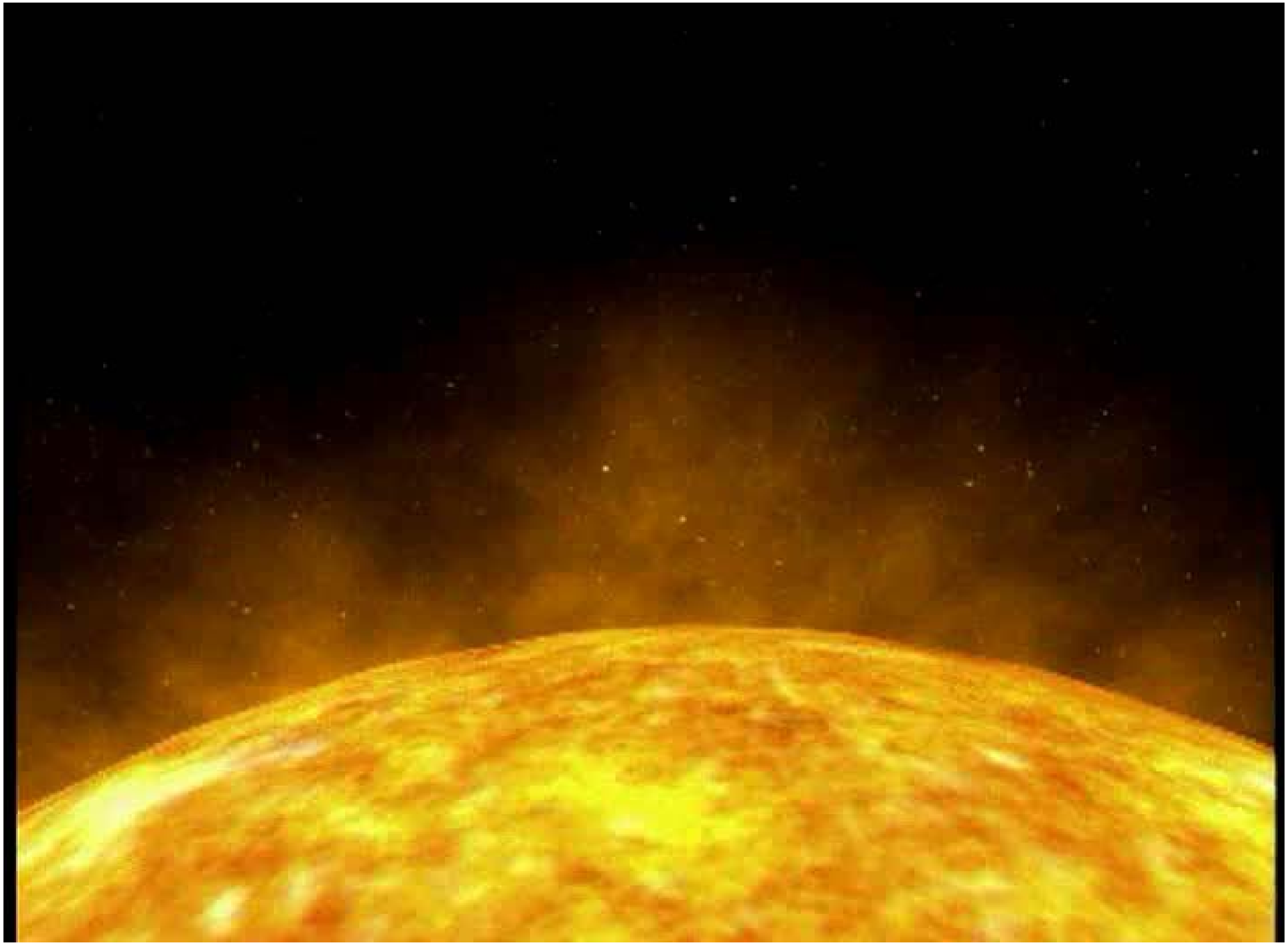




These are all limited by compute power

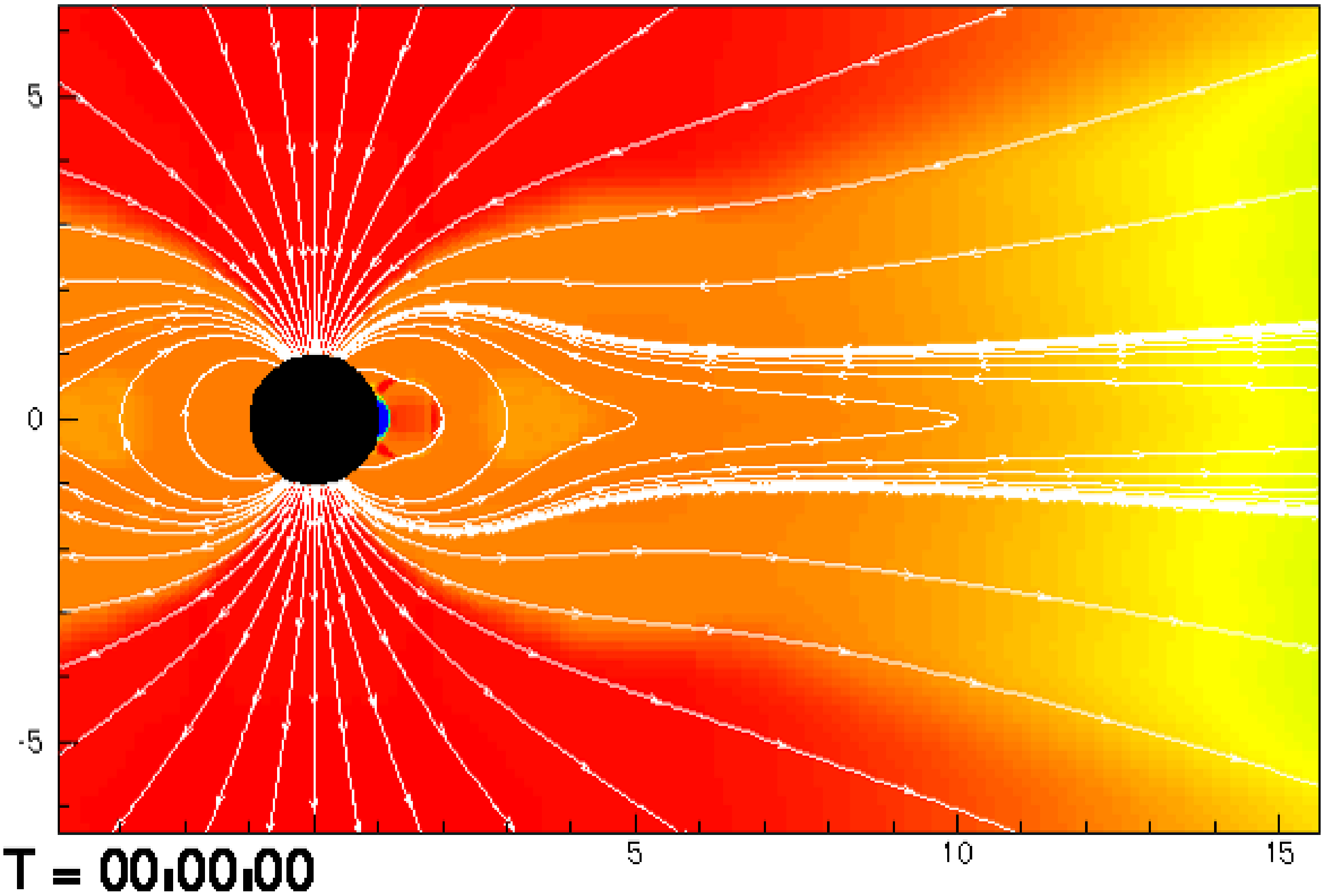
ExaScience Lab
Intel Labs Europe

SPACE WEATHER PREDICTION



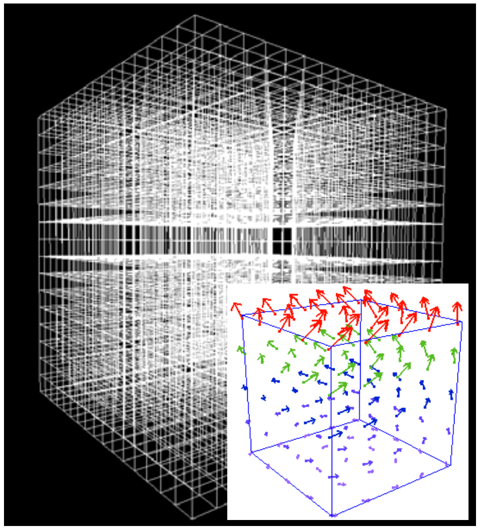
SPACE WEATHER PREDICTION

Center for Space Environment Modeling
University of Michigan

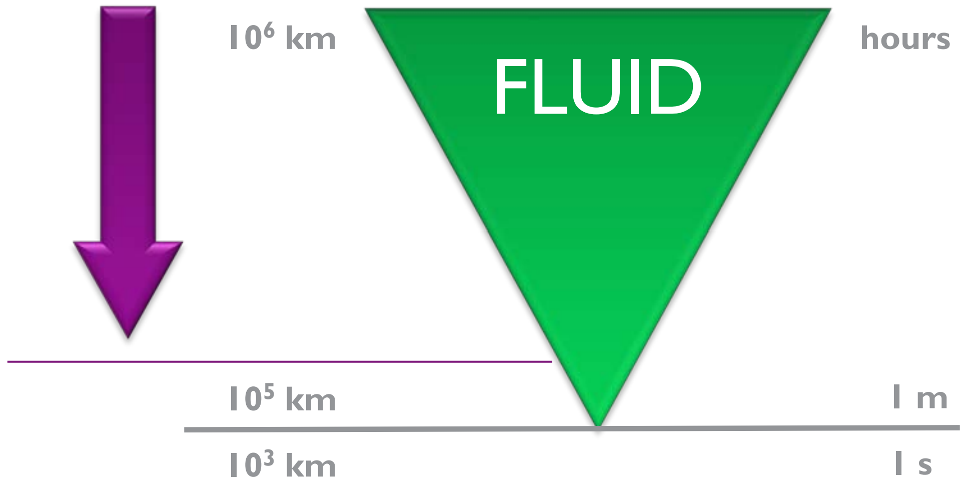


ExaScience Lab
Intel Labs Europe

SPACE WEATHER PREDICTION

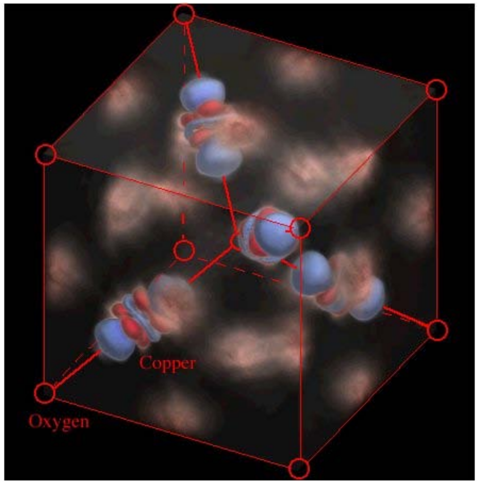


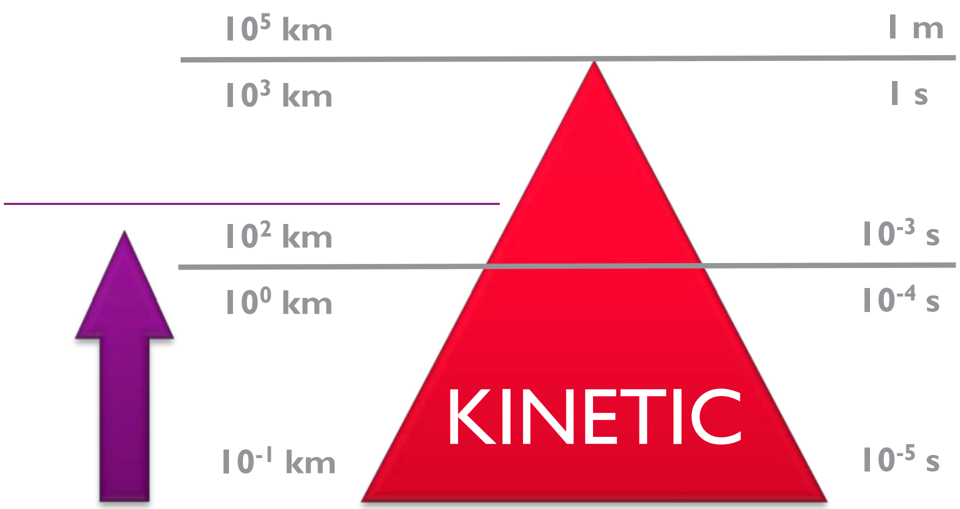
Petascale



SPACE WEATHER PREDICTION

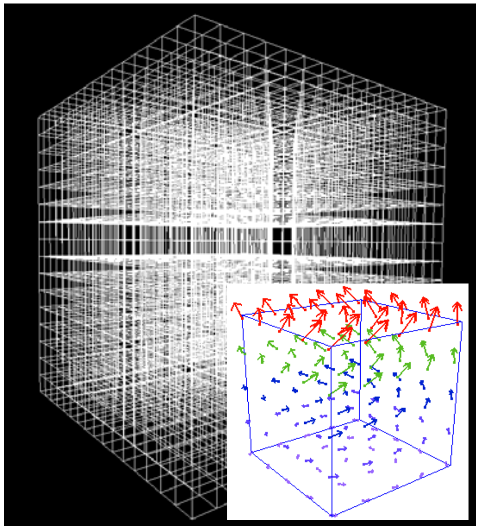
Petascale

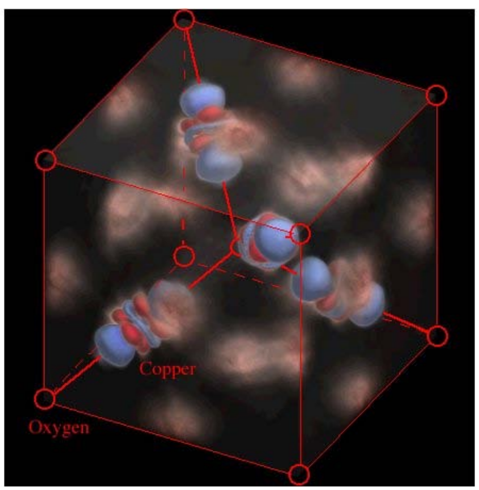


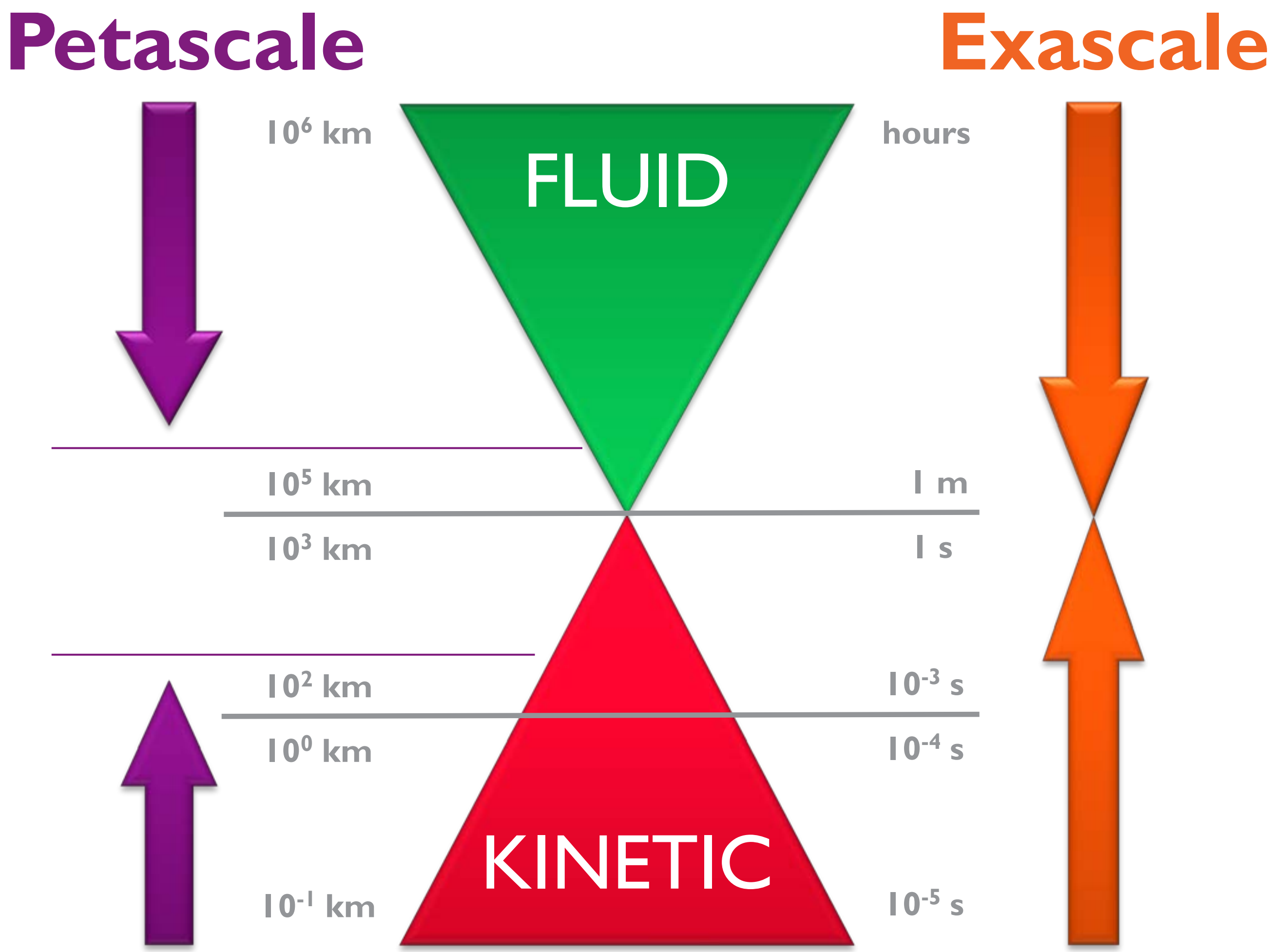


ExaScience Lab Intel Labs Europe

SPACE WEATHER PREDICTION





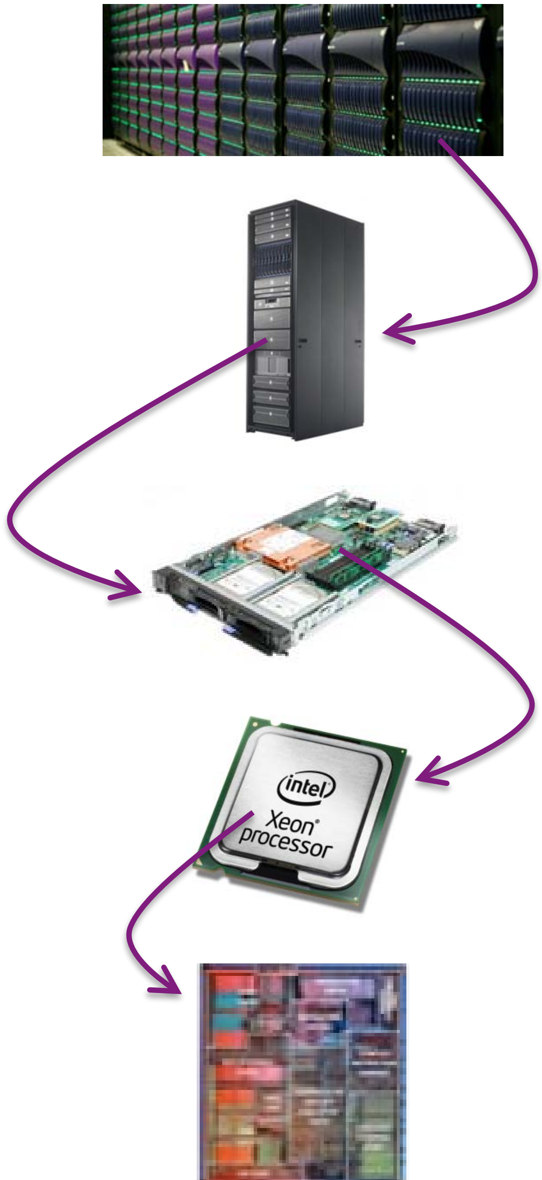


ExaScience Lab
Intel Labs Europe

Applications: computationally intensive, billions of threads

Hardware

INSIDE A HPC CENTER



Datacenter: 10^9 threads

Rack: 10^4-10^5 threads

Socket/blade: 500-5000 threads

Die: 100-1000 threads

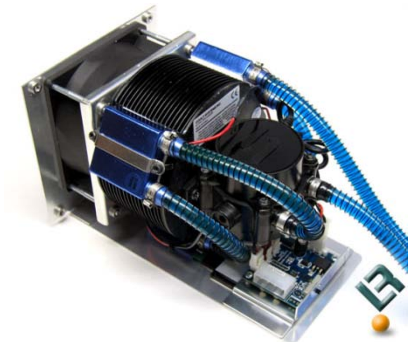
Core/tile: 1-10 threads

WHAT ARE THE CHALLENGES ?

energy, energy, energy

ExaScience Lab
Intel Labs Europe

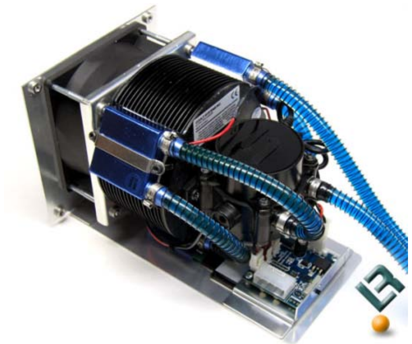
WHAT ARE THE CHALLENGES ?



energy, energy, energy

heat

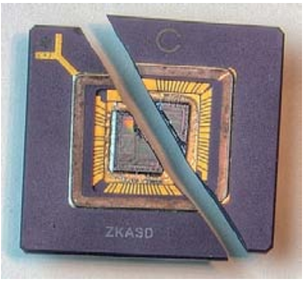
WHAT ARE THE CHALLENGES ?



energy, energy, energy

heat

hardware failures

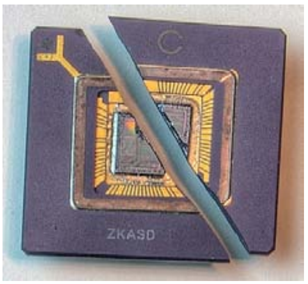


ExaScience Lab
Intel Labs Europe

WHAT ARE THE CHALLENGES ?

energy, energy, energy

heat



hardware failures

programming massively parallel machines
(a million cores, a billion threads!)

Applications: computationally intensive, billions of threads

?

Hardware: millions of cores, runtime variability and failures, energy

ExaScience Lab
Intel Labs Europe

KATHOLIEKE UNIVERSITEIT LEUVEN

UNIVERSITEIT GENT

Vrije Universiteit Brussel

Universiteit Antwerpen

universiteit hasselt
UNIVERSITEIT VAN DE TOEKOMST

Vlaamse overheid

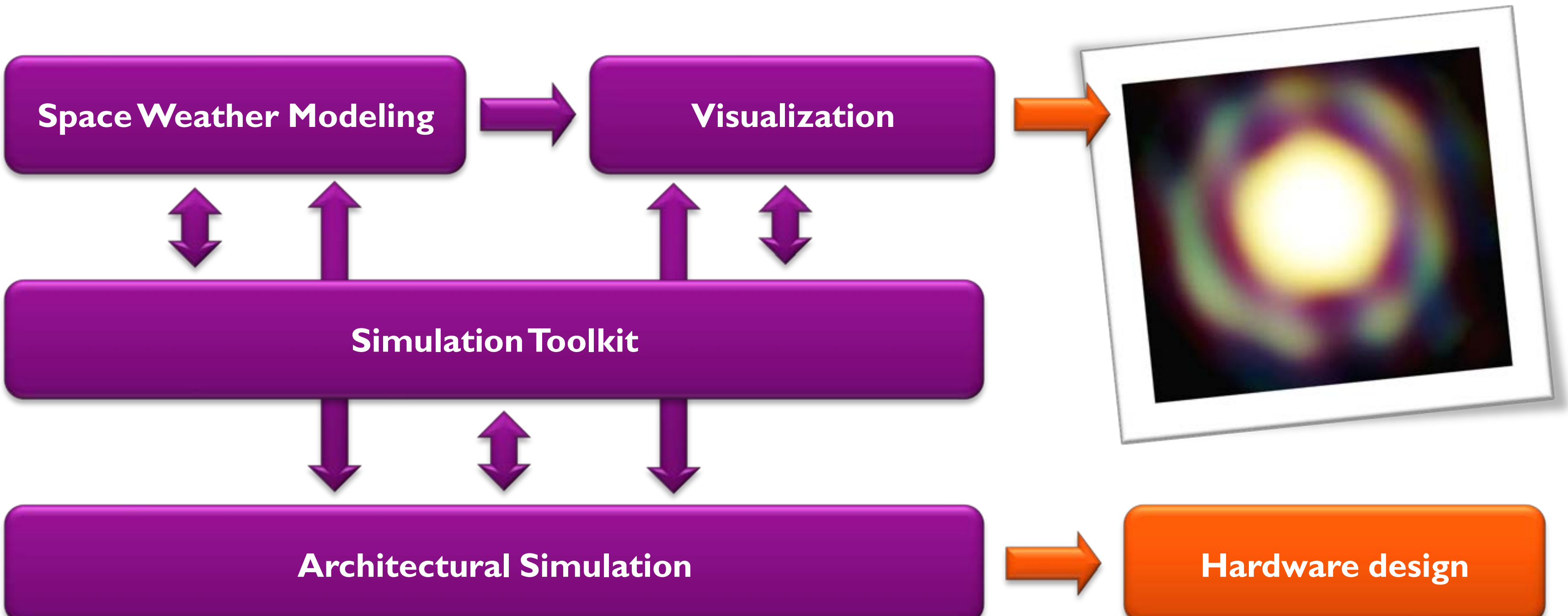
imec

intel

RESEARCH TOPICS

the ExaScience Lab will advance state of the art:

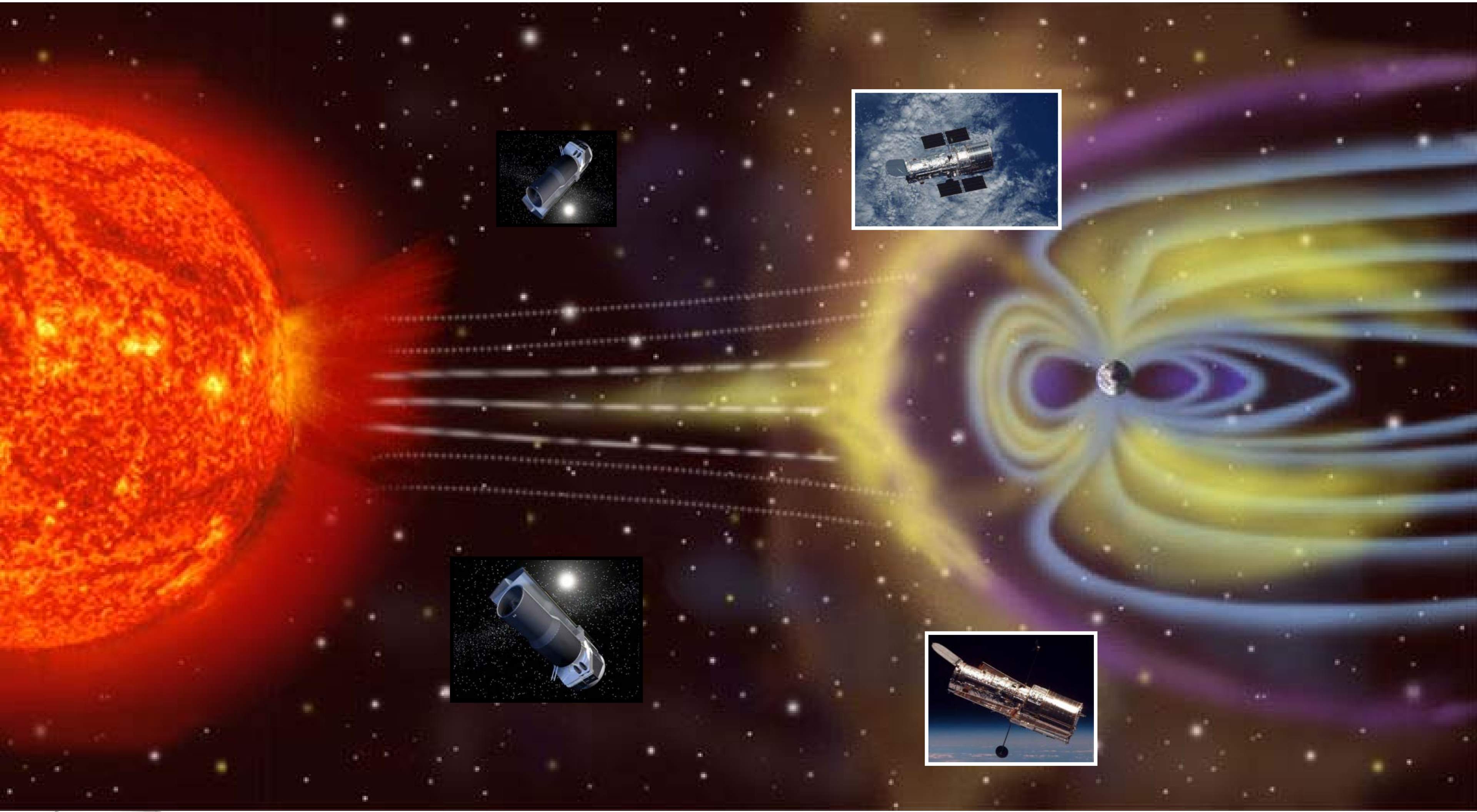
- More accurate space weather modeling and simulation
- Extremely scalable, fault tolerant simulation toolkit
- In-situ visualization through virtual telescopes
- Architectural simulation of large-scale systems and workloads



ExaScience Lab
Intel Labs Europe

VISUALIZATION

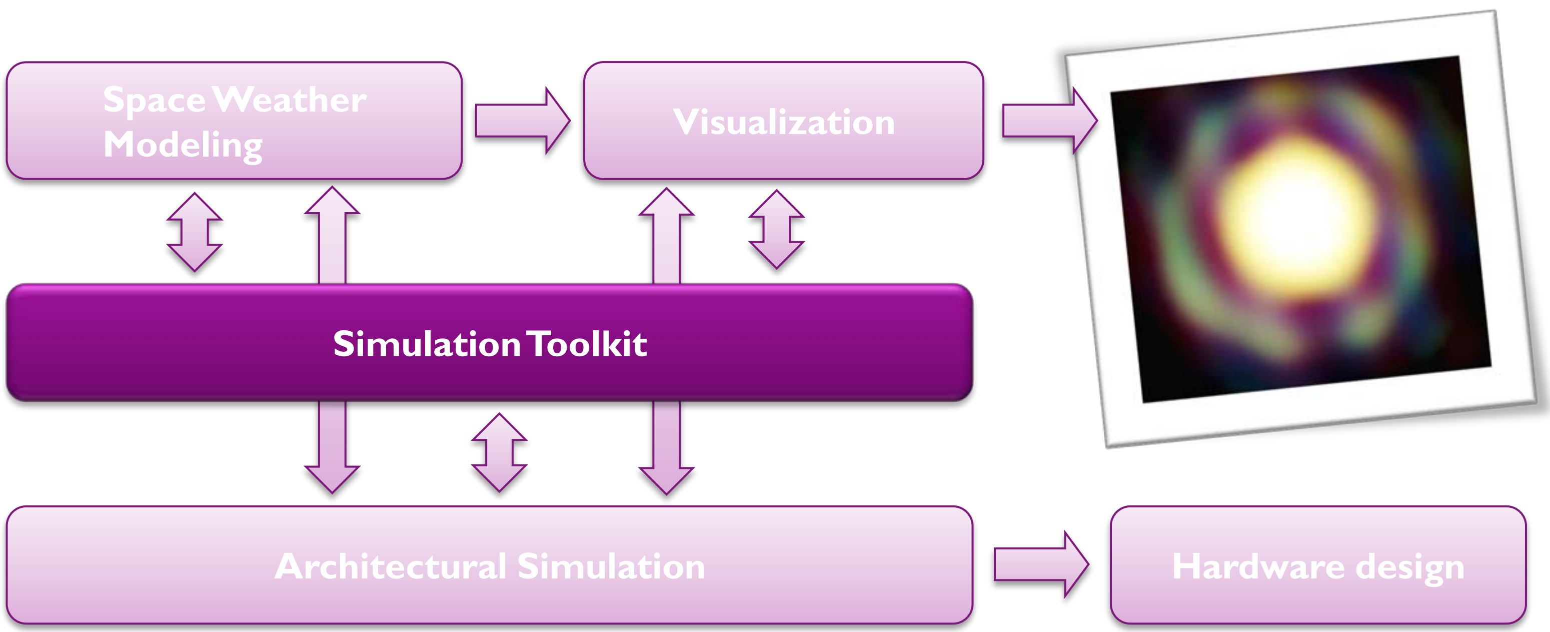
longer term objective = *Virtual Telescopes*



ExaScience Lab
Intel Labs Europe

SIMULATION TOOLKIT

HOW TO PROGRAM AN EXASCALE SYSTEM ?



ExaScience Lab
Intel Labs Europe

SIMULATION TOOLKIT

Isn't this just our current software programs x 1.000.000 ?

Extreme Parallelism

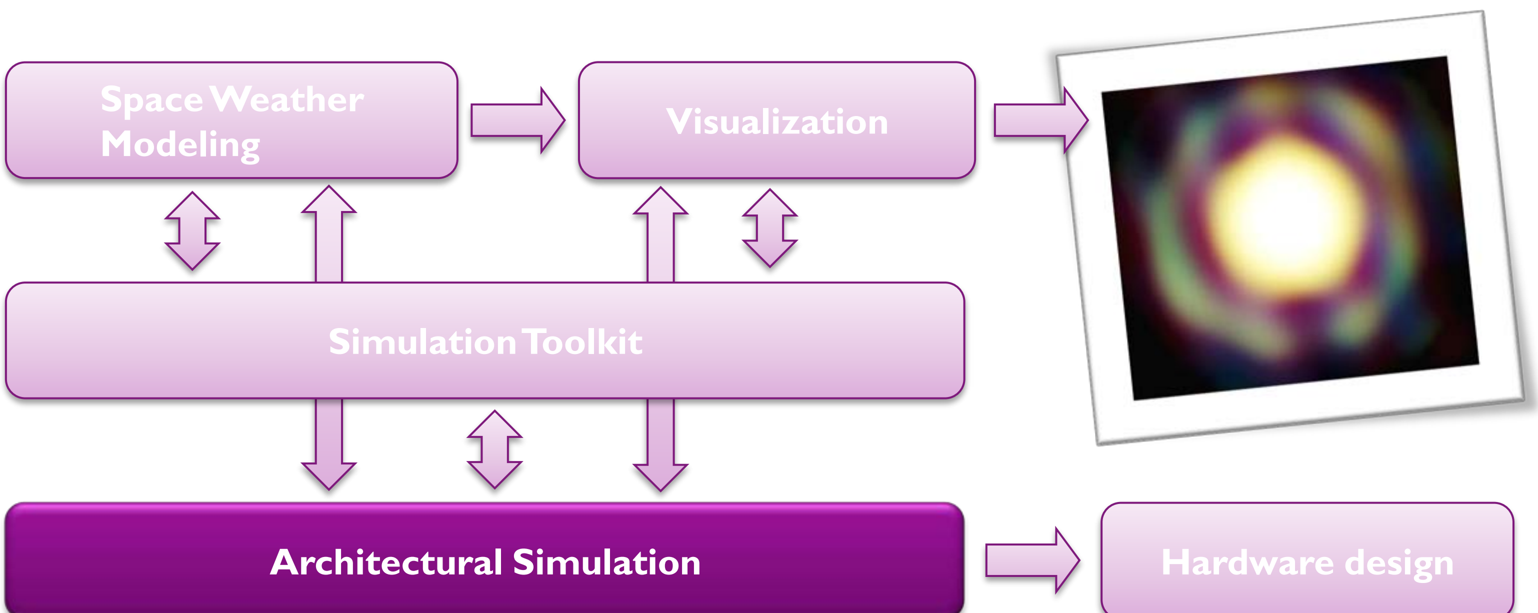
- Optimized implementations of 'ultra' scalable numerical kernels
- Heterogeneous Architecture
- Support for exascale parallel programming models

Dealing with hardware failures at the software level

- Dynamic load balancing
- Guarantee fault-tolerance

ARCHITECTURAL SIMULATION

HOW TO SIMULATE AN EXASCALE SYSTEM ?



ARCHITECTURAL SIMULATION

Before running a program on an exascale system, developers need to be able to predict:

- Performance
- Power
- Reliability
- Resource utilization

Usually this happens by simulating the system on a more powerful computer system.

- But this is going to be the biggest system on earth.

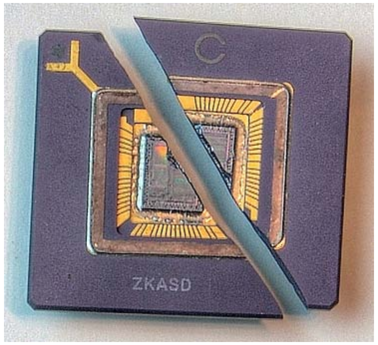
So, how will we be able to simulate this system?

ExaScience Lab
Intel Labs Europe

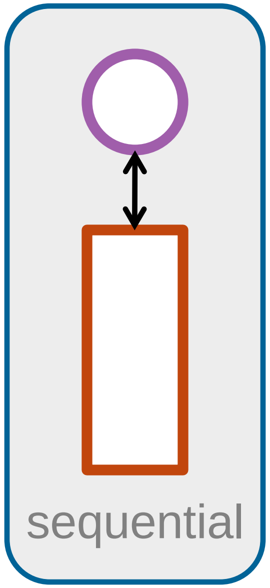
THE RUNTIME LAYER

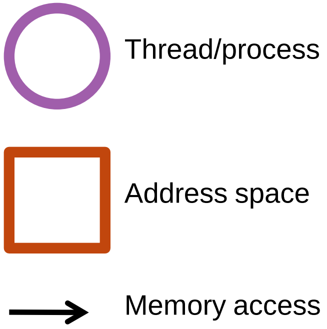
Software problems to tackle:

- Minimize energy consumption
  ... while remaining performant
- Deal with hardware variability and failures
  … transparently for the application
- Program massively parallel systems
  … without burdening the developer



PROGRAMMING MODELS





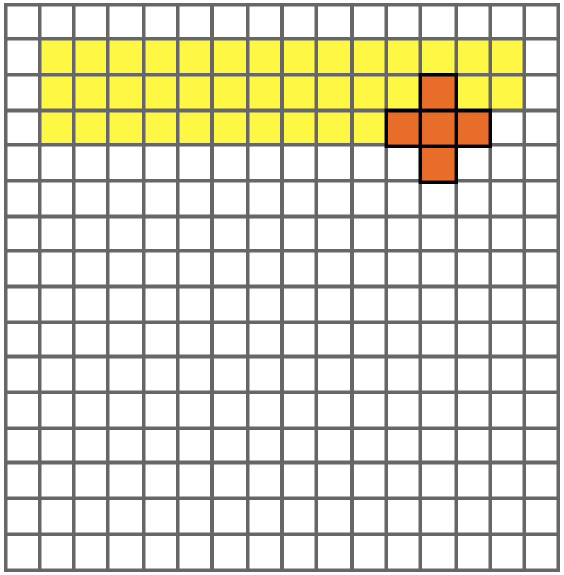
C, C++, Java (-threads), …

ExaScience Lab Intel Labs Europe

SEQUENTIAL PROGRAMMING EXAMPLE

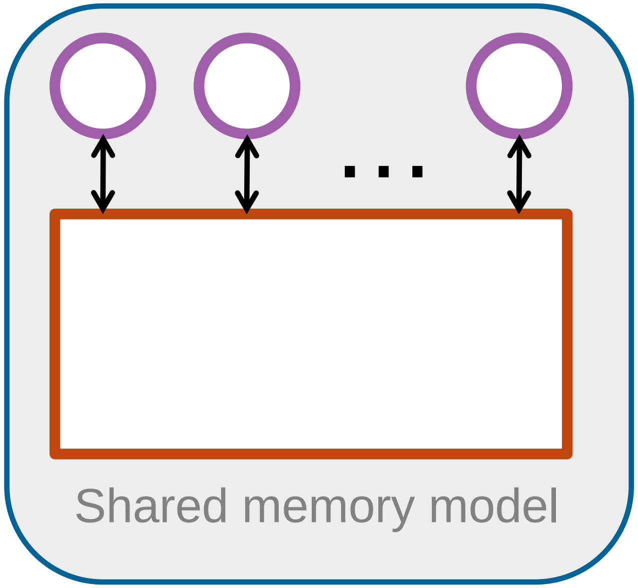
Explicit 2-dimensional heat distribution simulation

```
for(it=0; it<nr;it++)
        for(i=1; i<N; i++) {
                for(j=1; j<N; j++) {
                        ... stencil ...
                        (read from g1,
                         write to g2)
                }
        }
}
```

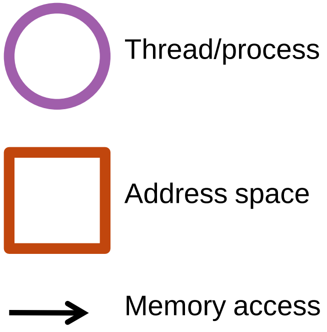


ExaScience Lab
Intel Labs Europe

PROGRAMMING MODELS



Java (+threads), Cilk, TBB, …



ExaScience Lab
Intel Labs Europe

SHARED MEMORY PROGRAMMING: CILK

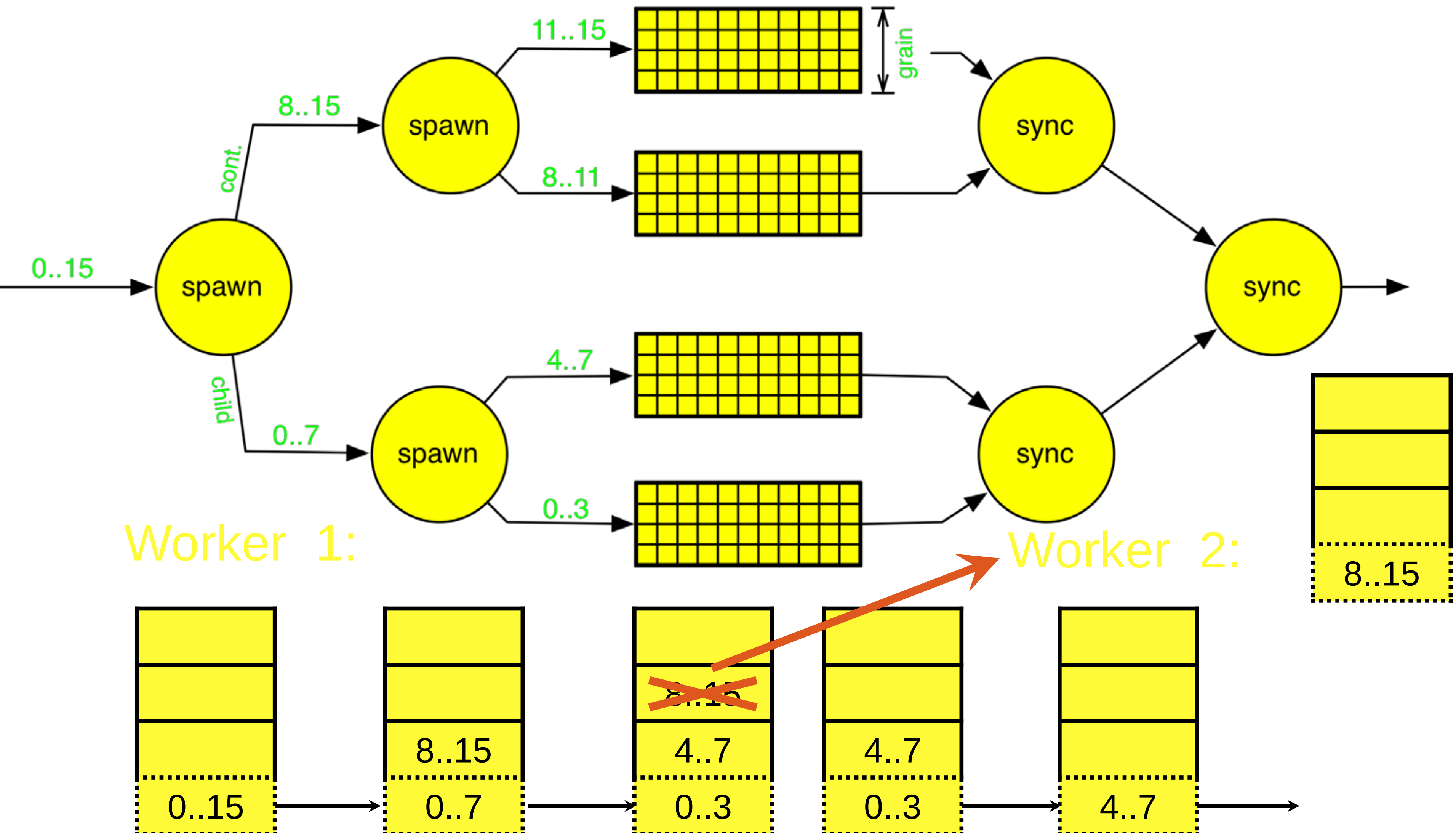
- New parallel function calling mechanism:
  - cilk_spawn indicates a call to function that can proceed in parallel with caller
  - cilk_sync awaits finish of spawned children
- work-stealing scheduler

```
#include <cilk/cilk.h>

…
int i, j;

cilk_for(i = 1; i < n; i++)
        for(j = 1; j < n; j++)
                v[P(i,j)] = …
```

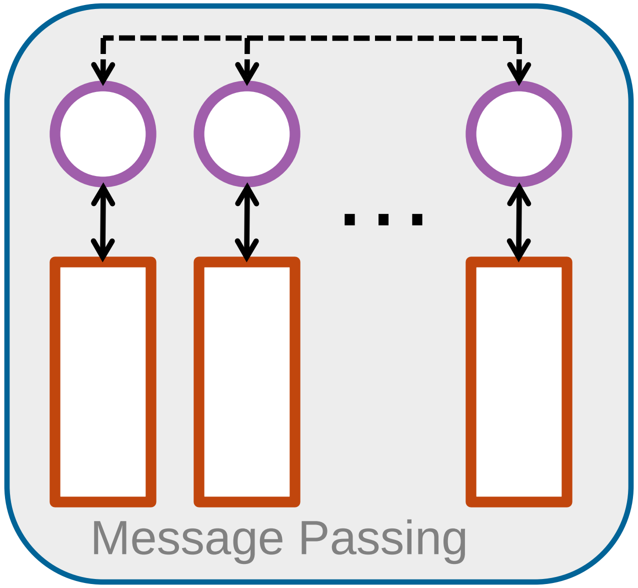
ExaScience Lab
Intel Labs Europe

WORK STEALING

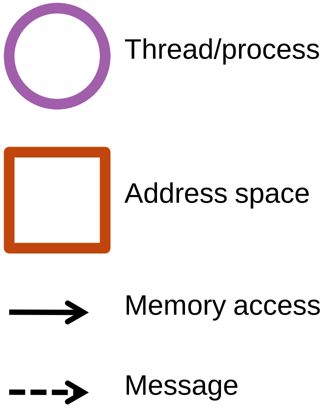
```
cilk_for(i = 1; i < n; i++)
        for(j = 1; j < n; j++)
                v[P(i,j)] = …
```



PROGRAMMING MODELS



MPI, actor models, …



ExaScience Lab
Intel Labs Europe

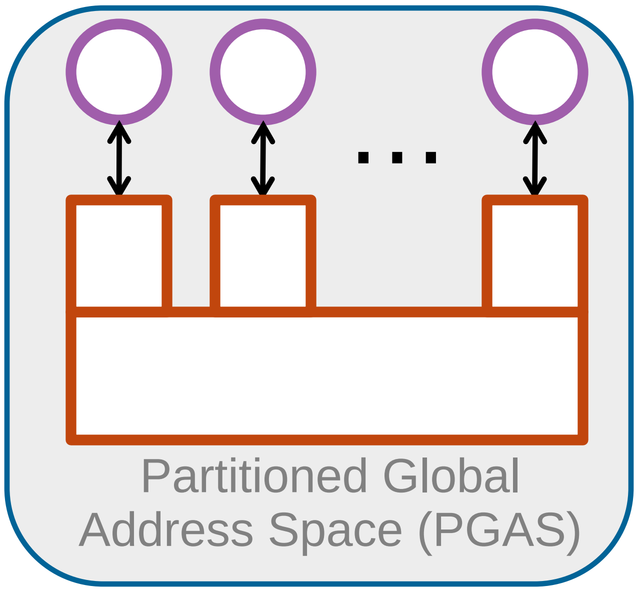
ACTORS IN SCALA

- Scala blends object-oriented and functional programming
  - Pure OO
  - Classes and inheritance
  - Block closures
  - Type inference
  - Higher-order functions
  - Sophisticated static type system

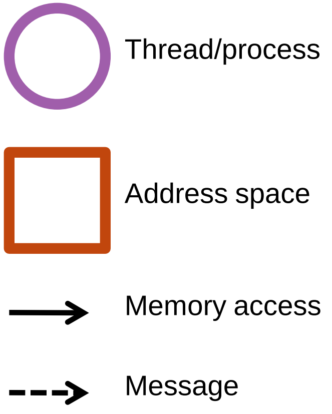
```
…
class Worker (signal : MailBox)

                  extends Runnable
{
   var func = () => () ;

   def run ()
   {
      var goOn = true;
      while (goOn) {
        signal.receive
        {
          case Go()   => func();
          case Stop() => goOn = false;
        };
        signal.send(Done());
      }
    }
}
…
```

ExaScience Lab
Intel Labs Europe

PROGRAMMING MODELS



UPC, Chapel, X10, …



UPC (UNIFIED PARALLEL C), A PGAS LANGUAGE

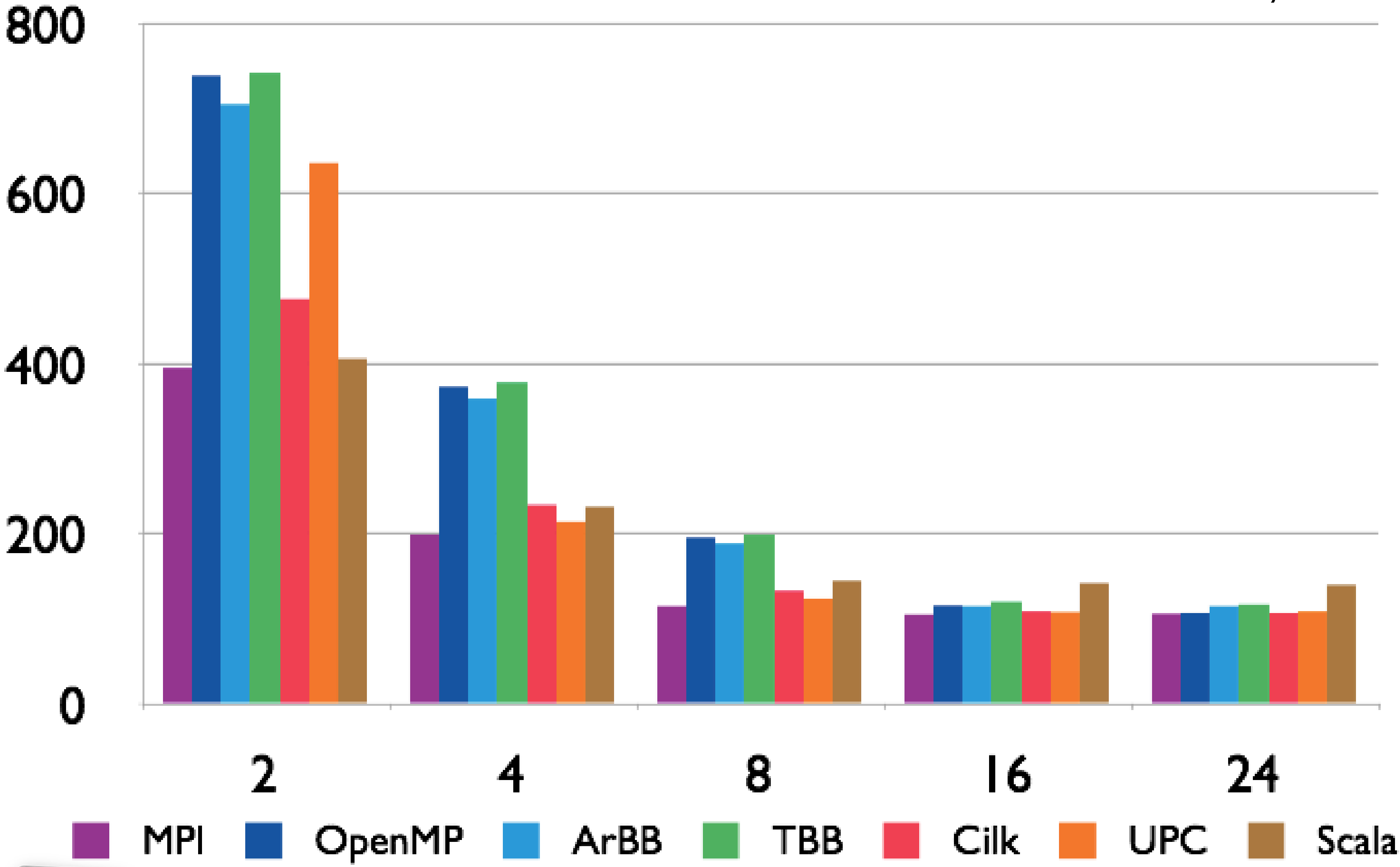
- Threads working in SPMD fashion
  - MYTHREAD specifies thread index
  - THREADS specifies number of threads (workers)
- shared keyword indicates shared scalars or arrays
  - Shared data has affinity to one thread
- Synchronization when needed (barriers, locks)

```
...
int i, j;

for(i = 1; i < n; i++)
  upc_forall(j = 1; j < n; j++; &v[i][j])
    v[i][j] = stencil(nu, u[i][j],
                          u[i][j-1],
                          u[i][j+1],
                          u[i-1][j],
                          u[i+1][j]);
...
```

ExaScience Lab
Intel Labs Europe

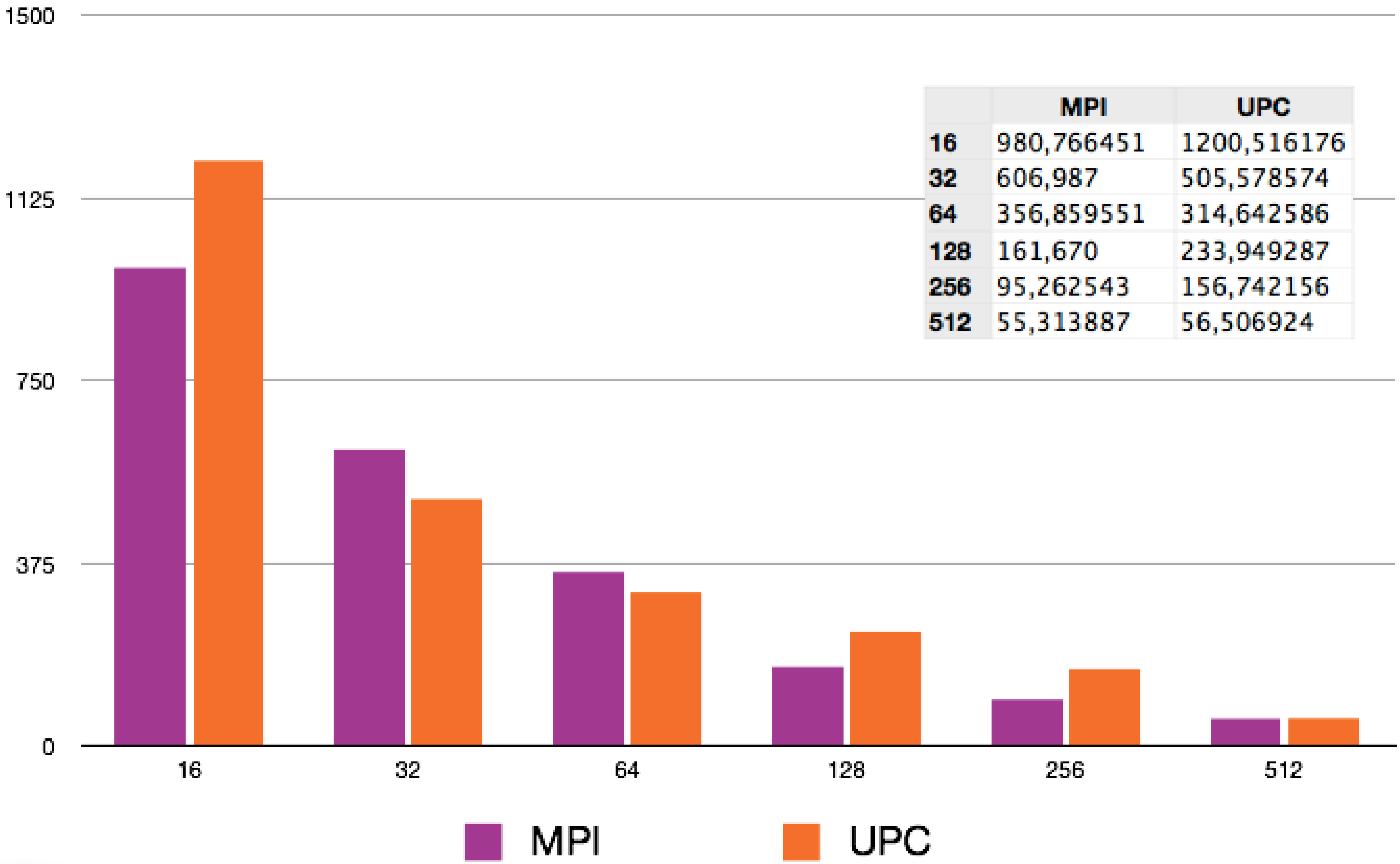
PERFORMANCE ON A SHARED MEMORY MACHINE

32768^2, 200it



PERFORMANCE ON SUPERCOMPUTER

Vic3, 32768^2, 2000it



	MPI	UPC
16	980,766451	1200,516176
32	606,987	505,578574
64	356,859551	314,642586
128	161,670	233,949287
256	95,262543	156,742156
512	55,313887	56,506924

ExaScience Lab
Intel Labs Europe

LESSONS LEARNED

UPC performs quite well

- But SPMD will not work on ExaScale machines
- Impact of memory affinity is huge

Work stealing shows promise

- Achieves automatic runtime load balancing with reasonable performance

Conclusion:
we want *memory affinity aware work stealing*

REACTIVE WORK REBALANCING EXPERIMENT ON CLUSTER

Start by giving each thread the same number of grid rows to process

Measure how long it takes each UPC thread to process these rows

If necessary:
React by transferring rows to other UPC thread

ExaScience Lab
Intel Labs Europe

REACTIVE WORK REBALANCING: KERNEL IMPLEMENTATION IN UPC

```
...
for (iteration = start; iter < start+nr; iteration ++) {
  upc_barrier;

  //potentially rebalance rows between threads
  rebalance(mgr, iteration);
  upc_barrier;
  adjustWork(mgr, *old, *new, iteration);
  incrementIterationCounter(mgr);

  //do a heat step and measure how long it takes
  upc_barrier;
  timeTaken = (*step)(mgr, iteration, *old, *new);

  //update the internal information
  update(mgr, iteration, timeTaken);

}
...
```

ExaScience Lab
Intel Labs Europe

REACTIVE WORK REBALANCING: KERNEL IMPLEMENTATION IN UPC

```
…
for (iteration = start; iter < start+nr; iteration ++) {
  upc_barrier;

  //potentially rebalance rows between threads
  rebalance(mgr, iteration);
  upc_barrier;
  adjustWork(mgr, *old, *new, iteration);
  incrementIterationCounter(mgr);

  //do a heat step and measure how long it takes
  upc_barrier;
  timeTaken = (*step)(mgr, iteration, *old, *new);

  //update the internal information
  update(mgr, iteration, timeTaken);

}
…
```

ExaScience Lab
Intel Labs Europe

REACTIVE WORK REBALANCING: ACTION!

Heat distribution simulation (16384x16384, 2000 it)

Every third thread (0, 3, …) straggles

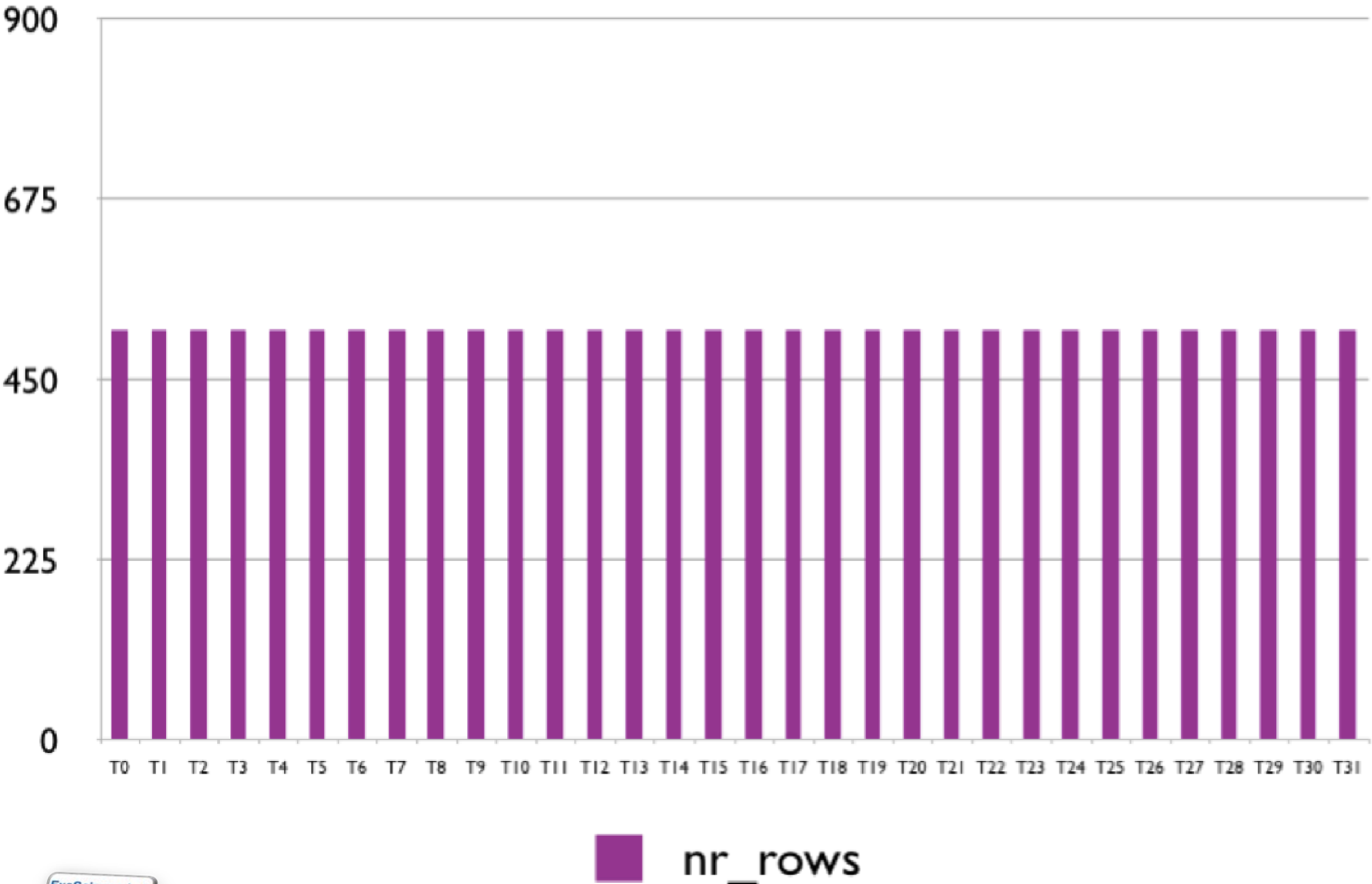
- slows down with a factor of 2

Ran distributed on 32 cores

X axis: core number

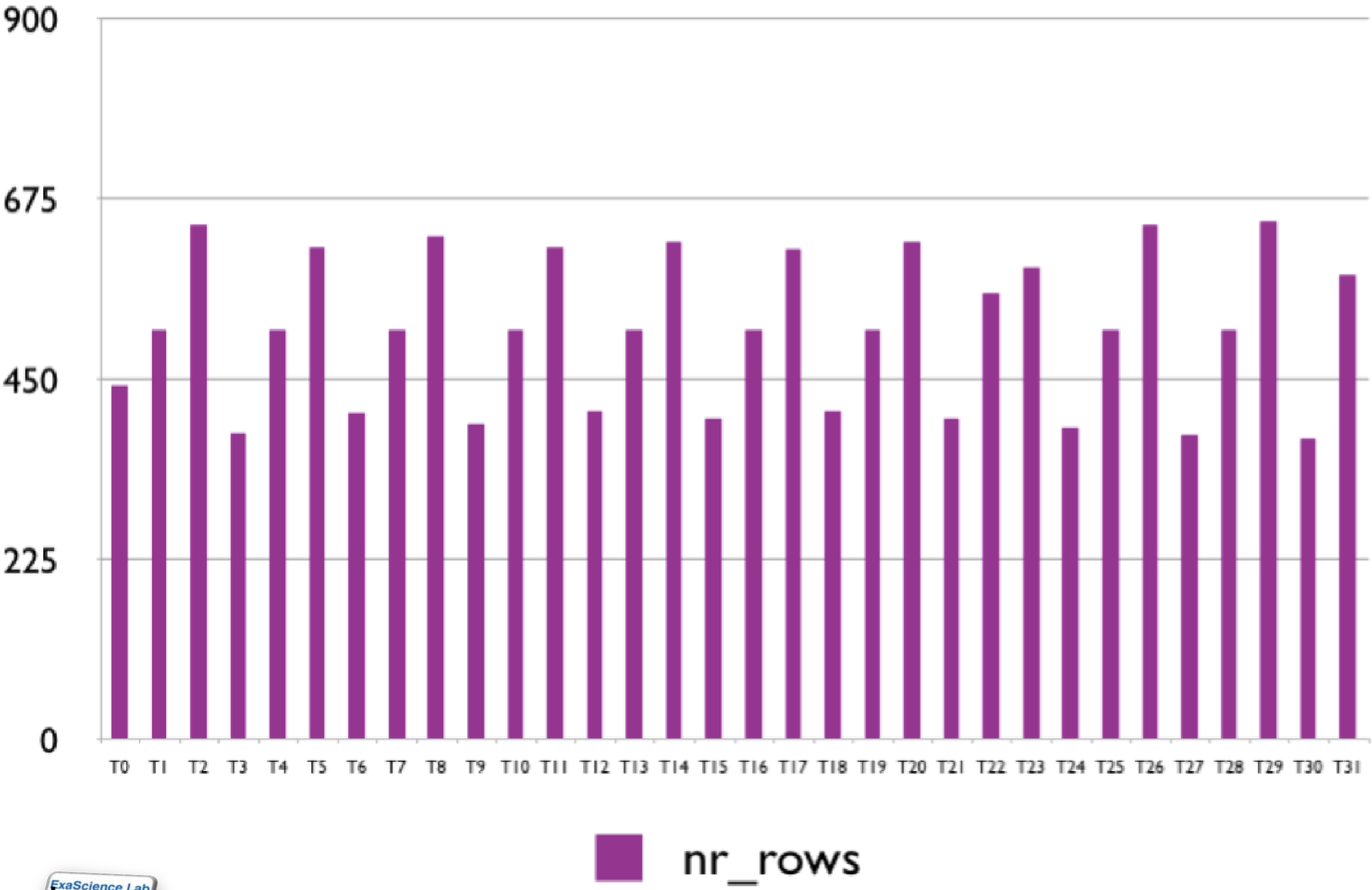
Y axis: nr of rows processed

ITERATION 0



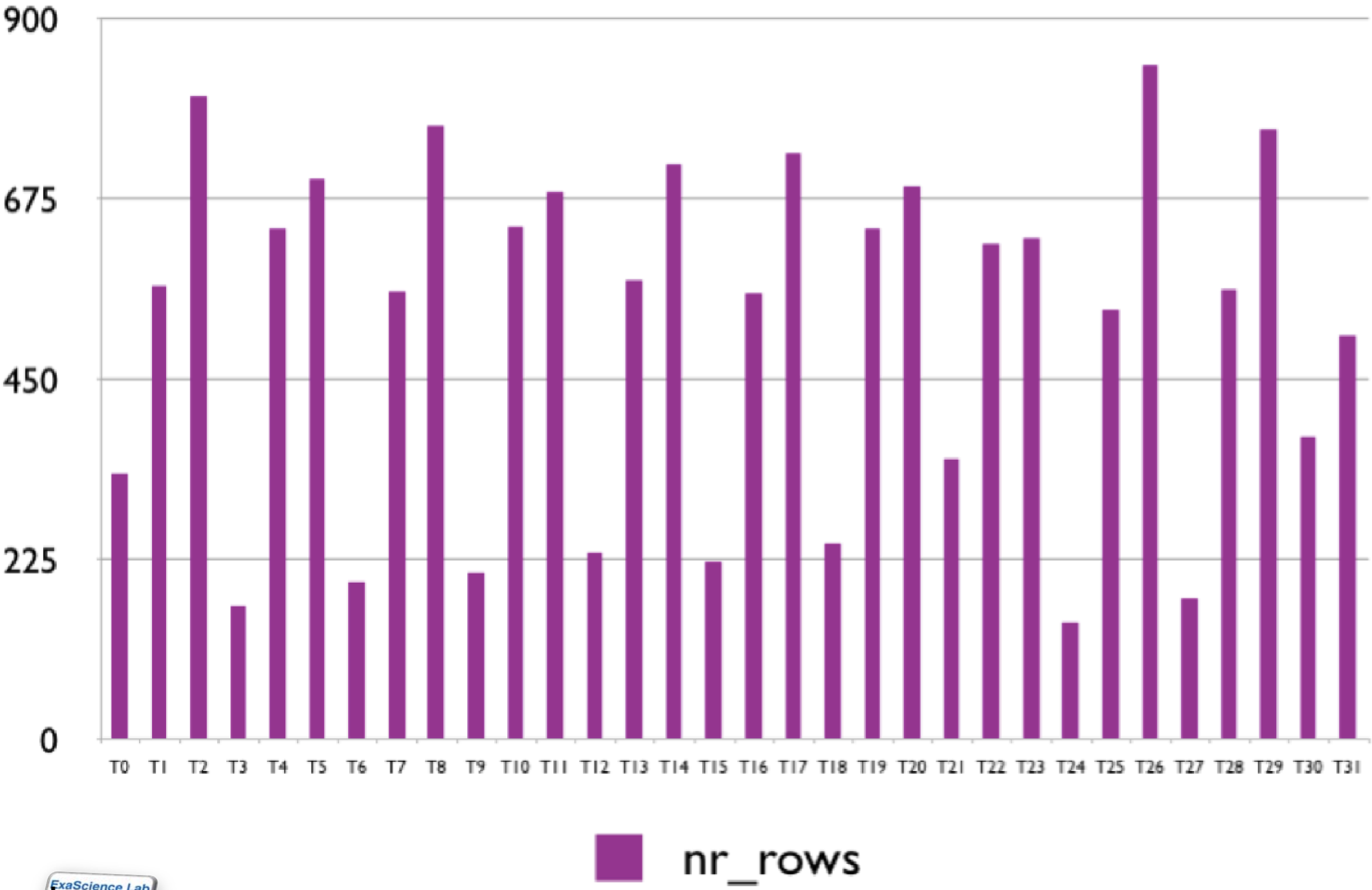
900
675
450
225
0
T0 T1 T2 T3 T4 T5 T6 T7 T8 T9 T10 T11 T12 T13 T14 T15 T16 T17 T18 T19 T20 T21 T22 T23 T24 T25 T26 T27 T28 T29 T30 T31
nr_rows

ITERATION 3



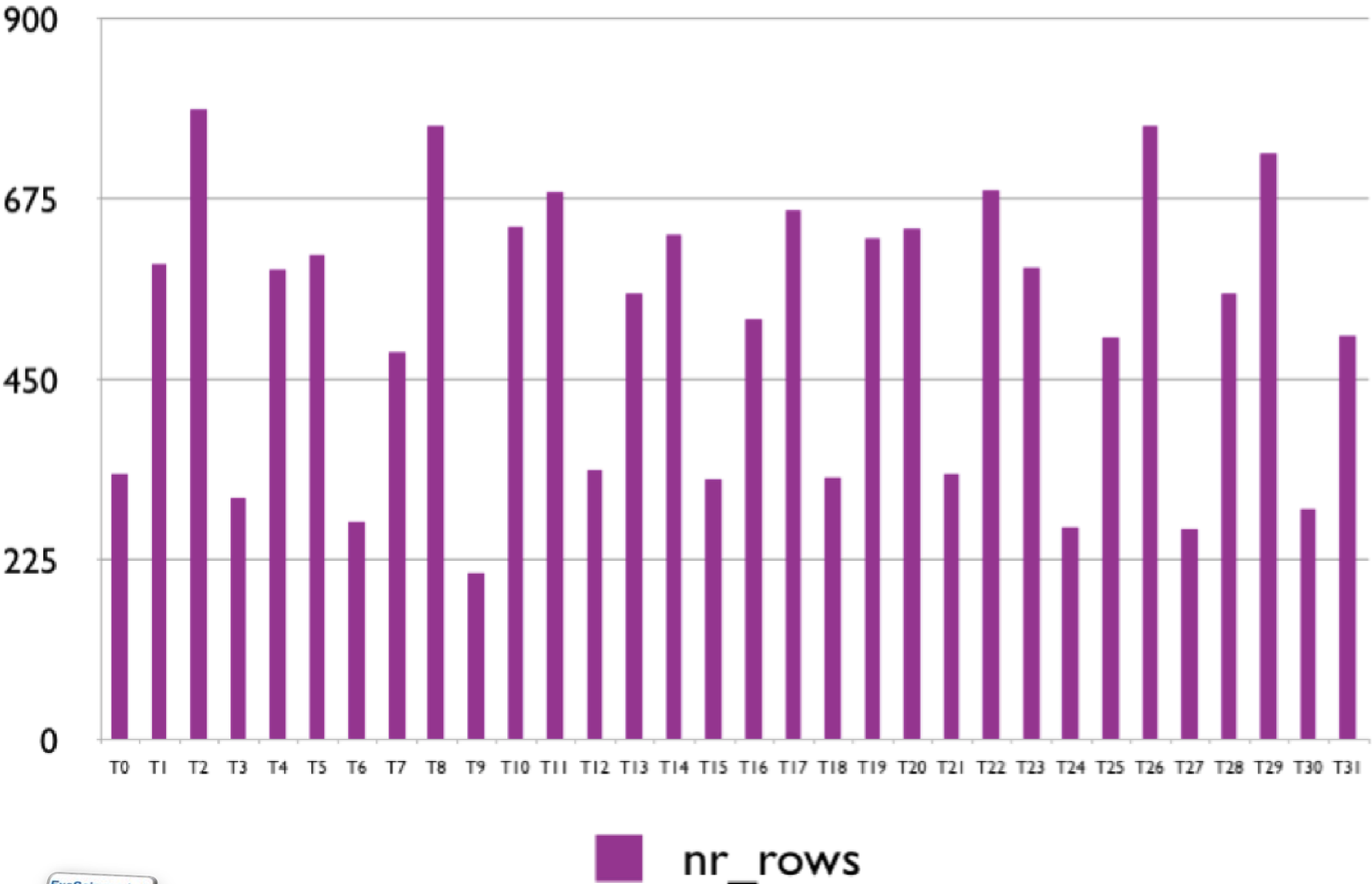
900
675
450
225
0
T0 T1 T2 T3 T4 T5 T6 T7 T8 T9 T10 T11 T12 T13 T14 T15 T16 T17 T18 T19 T20 T21 T22 T23 T24 T25 T26 T27 T28 T29 T30 T31
nr_rows

ITERATION 12

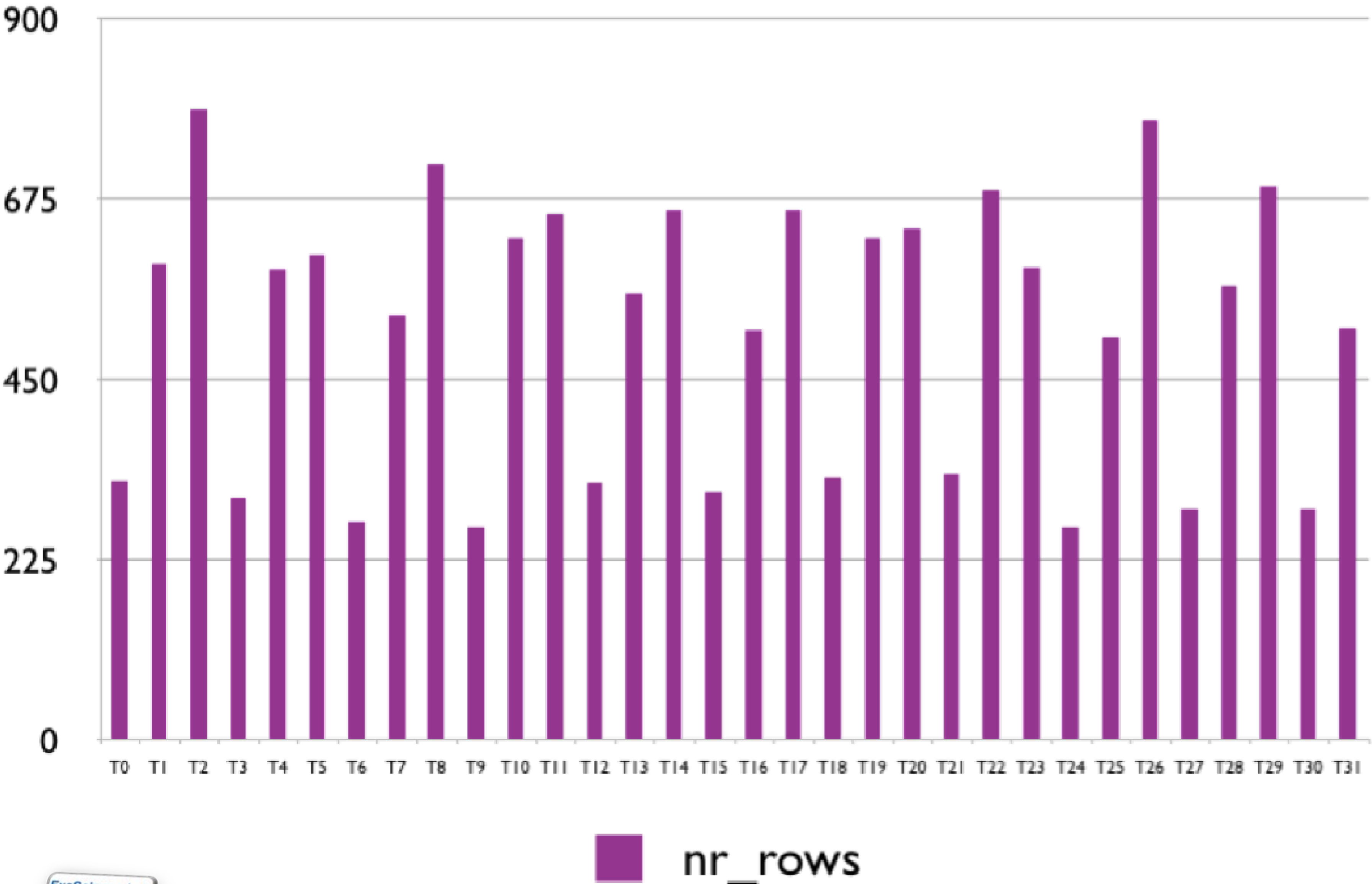


ITERATION 13

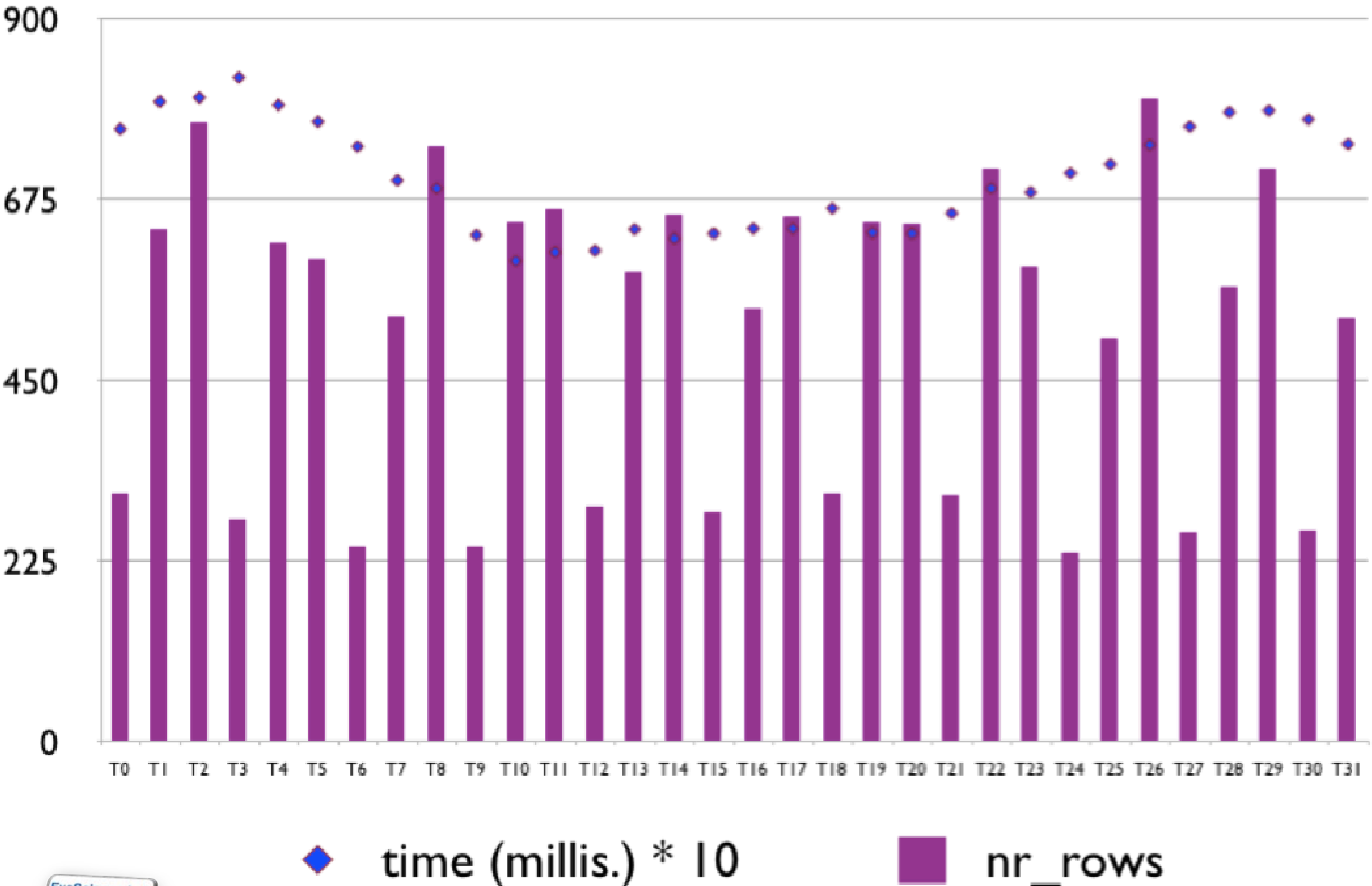
ITERATION 26



ITERATION 125

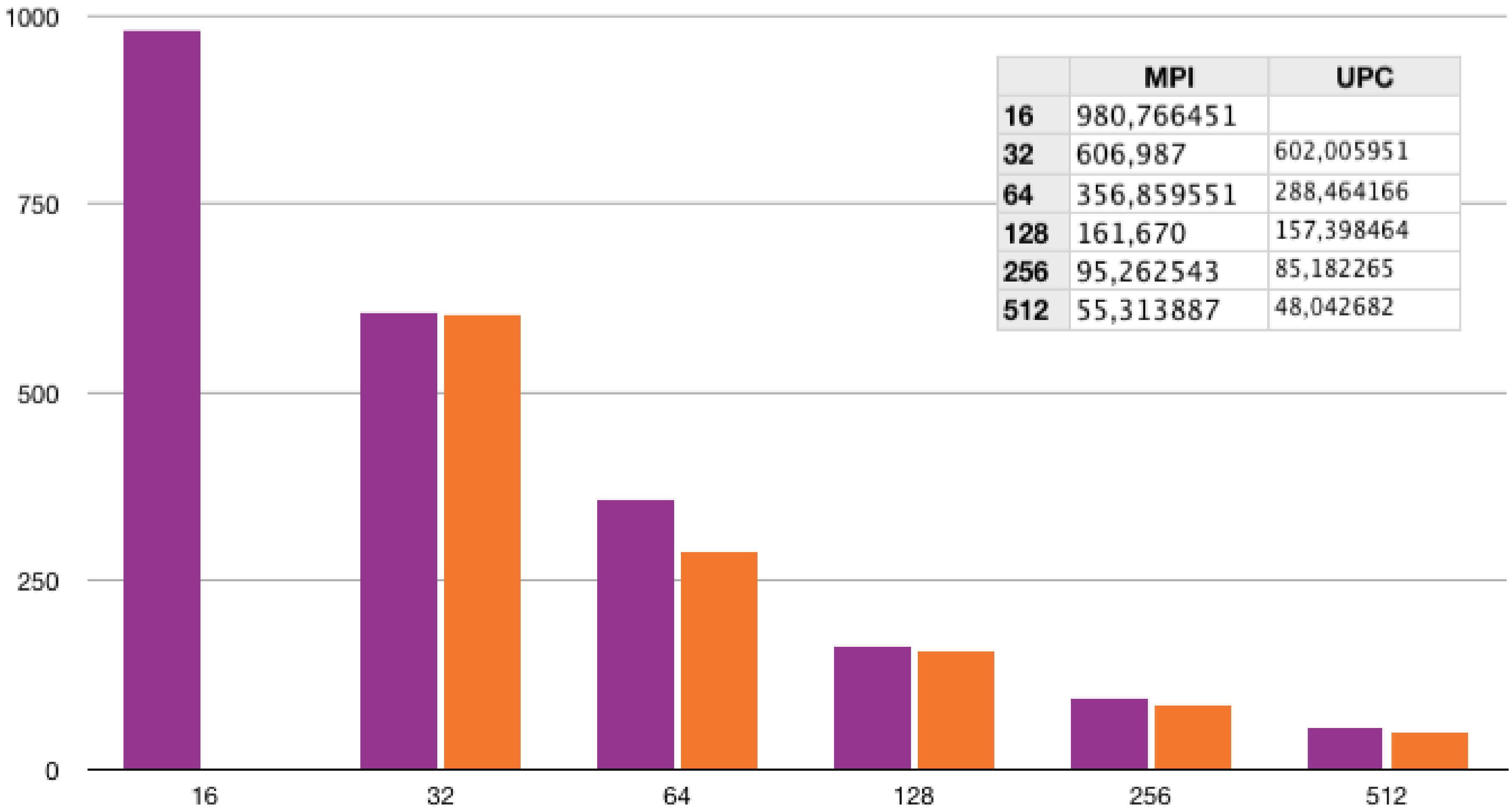


ITERATION 1999



PERFORMANCE ON SUPERCOMPUTER

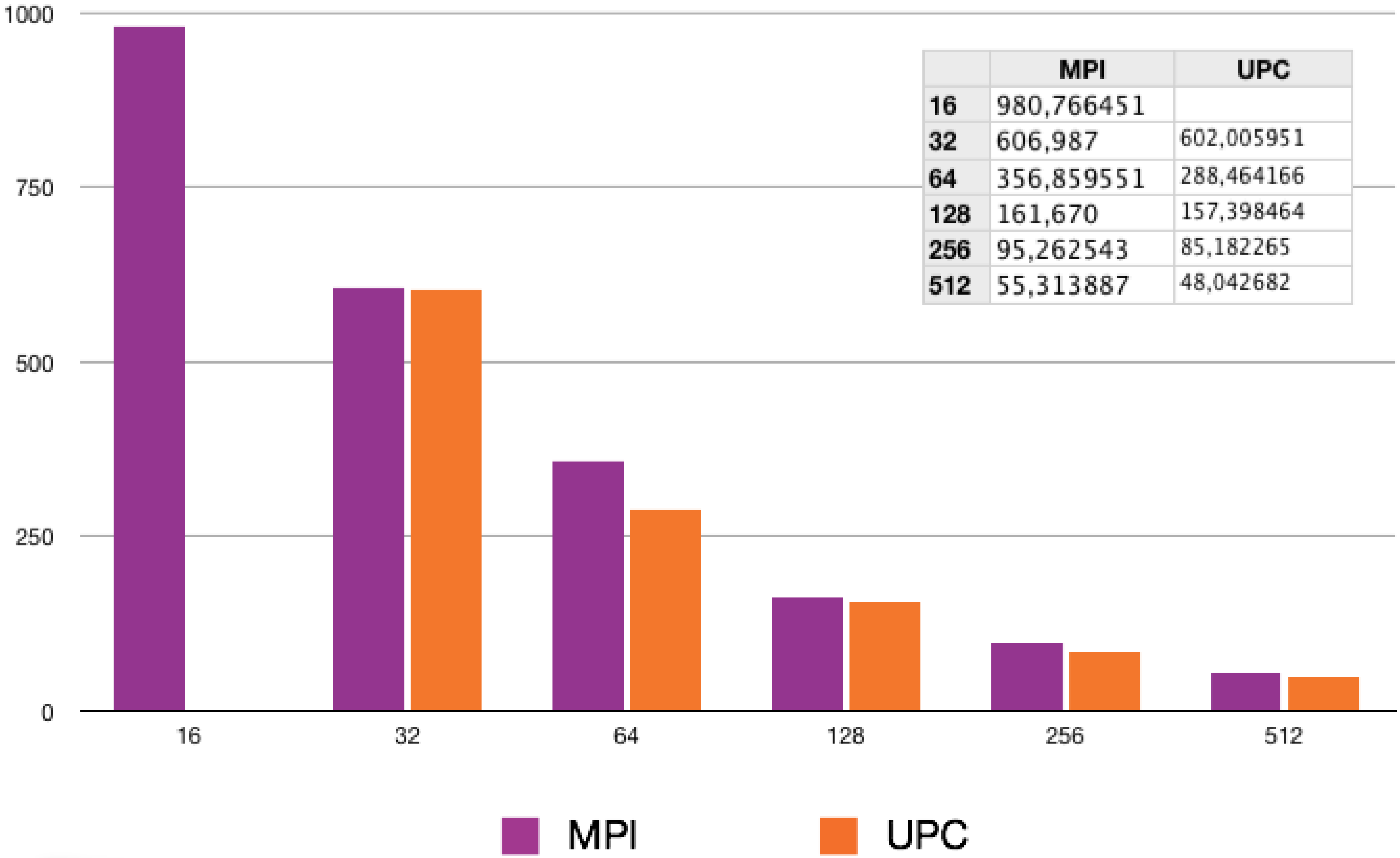
Vic3, 32768^2, 2000it



	MPI	UPC
16	980,766451	
32	606,987	602,005951
64	356,859551	288,464166
128	161,670	157,398464
256	95,262543	85,182265
512	55,313887	48,042682

PERFORMANCE ON SUPERCOMPUTER

Vic3, 32768^2, 2000it



	MPI	UPC
16	980,766451	
32	606,987	602,005951
64	356,859551	288,464166
128	161,670	157,398464
256	95,262543	85,182265
512	55,313887	48,042682

ExaScience Lab
Intel Labs Europe

Applications: computationally intensive, billions of threads

ExaScience Lab
Intel Labs Europe

Hardware: millions of cores, runtime variability and failures, energy

www.exascience.com

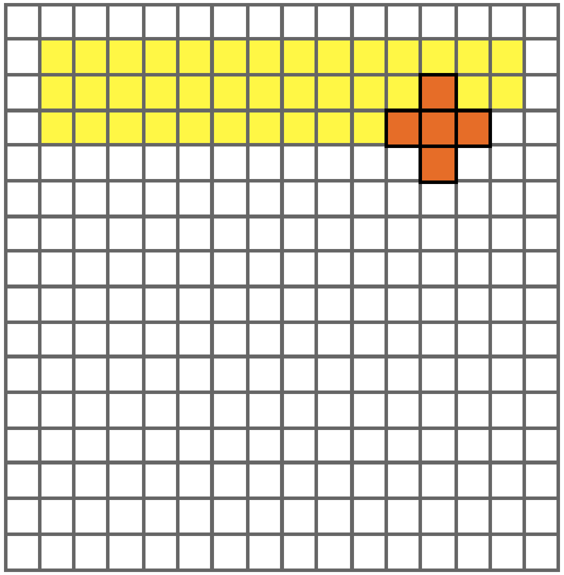
BACKUP SLIDES

ExaScience Lab
Intel Labs Europe

2D HEAT

- Grid
- Stencil Operation
- Explicit Timestepping

```
for(it=0; it<nr;it++)
        for(i=1; i<N; i++) {
                for(j=1; j<N; j++) {
                        ... stencil ...
                        (read from g1, write to g2)
                }
        }
}
```



ExaScience Lab
Intel Labs Europe

CILK

- New parallel function calling mechanism:
  - cilk_spawn indicates a call to function that can proceed in parallel with caller
  - cilk_sync awaits finish of spawned children
- work-stealing scheduler

```
#include <cilk/cilk.h>

…
int i, j;

cilk_for(i = 1; i < n; i++)
        for(j = 1; j < n; j++)
                v[P(i,j)] = …
```

ExaScience Lab Intel Labs Europe

TBB (THREADING BUILDING BLOCKS)

- C++ Threading library (Intel)
- Task based parallelism
- Task scheduler
- Work stealing approach similar to Cilk

- Automatic parallellization
  - parallel_for
  - reduce/scan/sort/while
- Parallel data structures
- Scalable memory allocation
- Mutual exclusion
- Atomic operations

```
#include <tbb/parallel_for.h>
#include <tbb/blocked_range.h>

thread_code(blocked_range &r){
  for (int j=r.begin(); j!=r.end(); j++)
     // Stencil on rows j
}


...
blocked_range r(0, numThreads);
parallel_for(r, thread_code, aff_part);
...
```

ExaScience Lab
Intel Labs Europe

ARBB (ARRAY BUILDING BLOCKS)

- C++ language extension for vector parallel programming
- Intel product
    - currently beta 2
- Computational kernel defined for the individual grid points
- Automatic parallelization over cores and SIMD units
- Can be extended to GPU
- Vector code is JIT-compiled

```
#include <arbb.hpp>

template<typename T>
void heat_stencil(T src, T& dst) {
    dst = src + D * (-4*src
                    + neighbor(src, -1, 0)
                    + neighbor(src, 1, 0)
                    + neighbor(src, 0, -1)
                    + neighbor(src, 0, 1));
}

template<typename T>
void heat_driver(dense<T, 2> grid,
                 dense<T, 2> next) {
    map(heat_stencil<T>)(grid, next);
}

int main(){
    ...
    call(heat_driver<T>)(grid, next);
    ...
}
```

ExaScience Lab
Intel Labs Europe

UPC

- Threads working in SPMD fashion
  - MYTHREAD specifies thread index
  - THREADS specifies number of threads (workers)
- shared keyword indicates shared scalars or arrays
  - Shared data has affinity to one thread
- Synchronization when needed (barriers, locks)

```
…
int i, j;

for(i = 1; i < n; i++)
  upc_forall(j = 1; j < n; j++; &v[i][j])
    v[i][j] = stencil(nu, u[i][j],
                                                u[i][j-1],
u[i][j+1],
                                                u[i-1][j],
u[i+1][j]);
…
```

ExaScience Lab
Intel Labs Europe

CHAPEL

- “Global View” paradigm - clean algorithm code
  - No communication/data sharing code intermixed
  - No explicit decomposition of data structures and control flow into per-task or per “node” chunks
- Data parallel features based on domains (index set)
- Data distribution customizable by introducing domain maps
  - separate from algorithm code
- Explicit Task parallellism

```
const MatrixSpace :
            domain(2) = [0..size, 0..size];
const  ProblemSpace :
            subdomain(MatrixSpace) = [1..n, 1..n];

def  heat(n, u, v, nu) {
 forall (i,j) in ProblemSpace do
     v(i,j) = u(i,j) + nu * ( u(i+1,j) …
}
```

X10

- Type-safe parallel OO language
- Asynchronous PGAS
  - data locality through places
  - lightweight activities

```
public class HeatTransfer_v1 {
  const BigD =
          Dist.makeBlock([0..n+1, 0..n+1], 0);
  const D = BigD | ([1..n, 1..n] as Region);
  const A = DistArray.make[Real]
             (BigD,(p:Point) =>{ …init…});
…
def run() {
  var iter:Int = 0;
  do {
   iter = iter + 1;
   finish ateach (p in D) Temp(p) = stencil(p);
   finish ateach (p in D) A(p) = Temp(p);
  } while (iter < iterations);
}
```

ExaScience Lab
Intel Labs Europe

SCALA

- Blends object-oriented and functional programming
  - Pure OO
  - Classes and inheritance
  - Block closures
  - Type inference
  - Higher-order functions
  - Sophisticated static type system

```
…
class Worker (signal : MailBox)

                  extends Runnable
{
    var func = () => () ;

    def run ()
    {
        var goOn = true;
        while (goOn) {
          signal.receive
          {
            case Go()   => func();
            case Stop() => goOn = false;
          };
          signal.send(Done());
        }
     }
}
…
```

ExaScience Lab
Intel Labs Europe

HPC VERSUS…

Grid/Cloud Computing

- loosely coupled heterogeneous systems that are geographically dispersed
- Computing resources are not administered centrally
- Interconnected through slower networks
- Typically runs several different programs in parallel