Barcelona Supercomputing Center
Centro Nacional de Supercomputación

EXCELENCIA SEVERO OCHOA

From the latency to the throughput age

Prof. Jesús Labarta

BSC & UPC

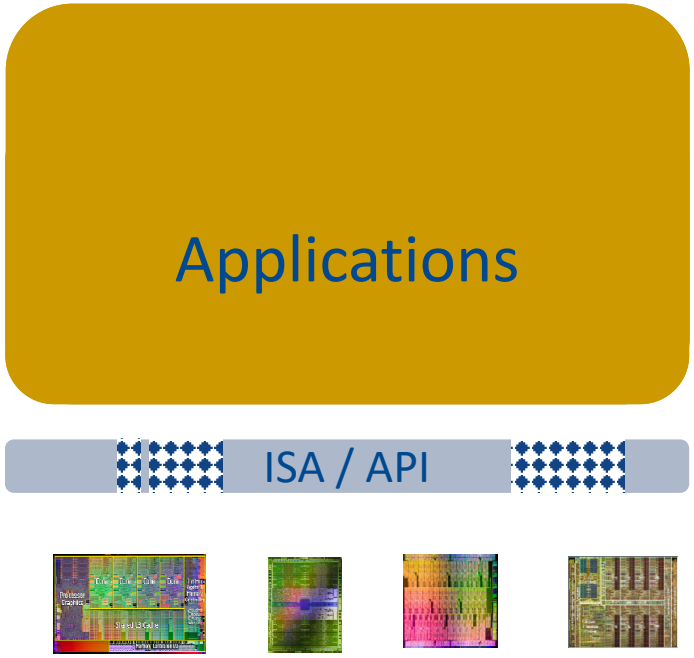
UCM

Madrid, November 26th 2018

Vision

- The multicore and memory revolution
  - ISA leak …
  - Plethora of architectures
    - Heterogeneity
    - Memory hierarchies

The power wall made us go multicore and the ISA interface to leak
→ our world is shaking

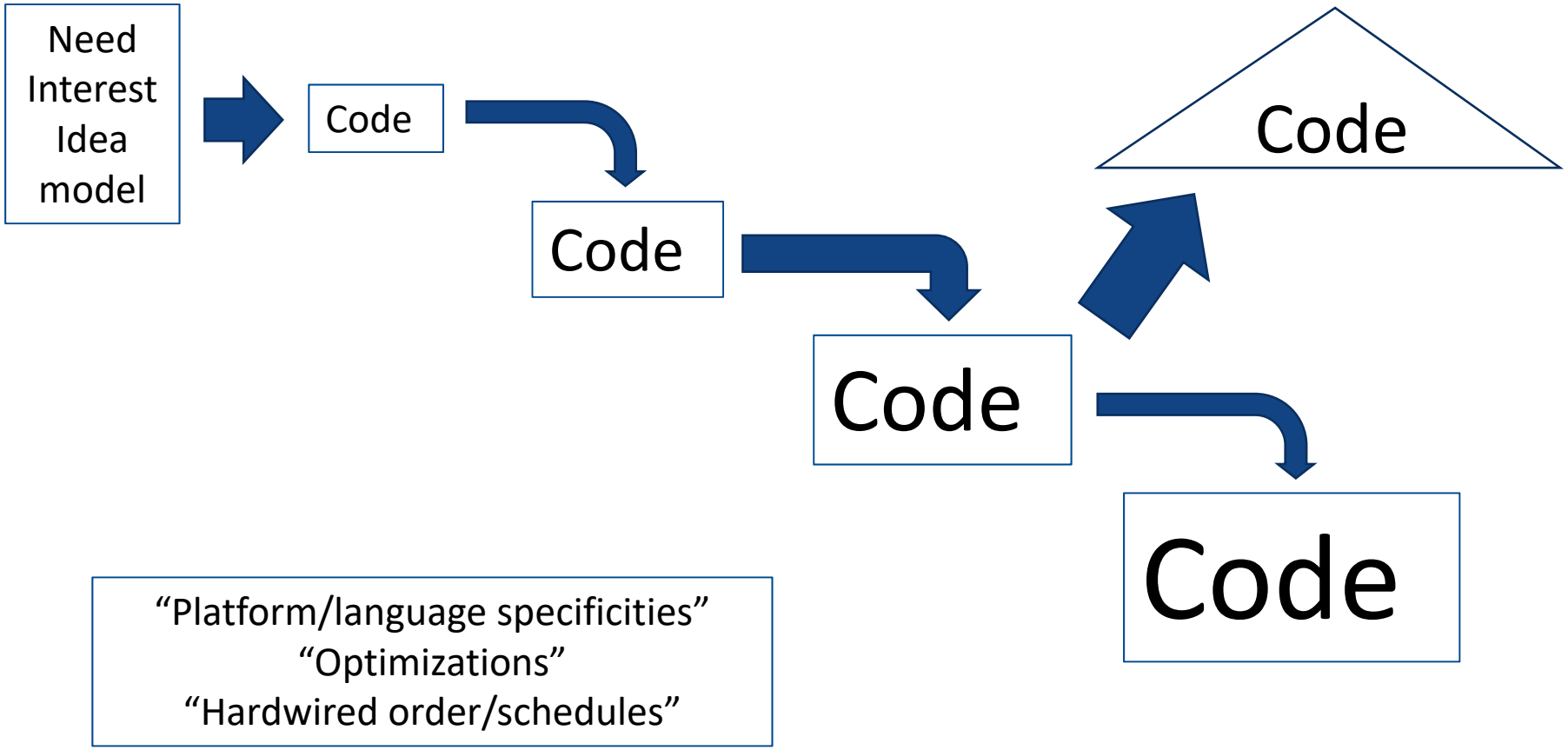


- Complexity + variability = Divergence
  - Between our mental models and actual system behavior

What programmers need ?

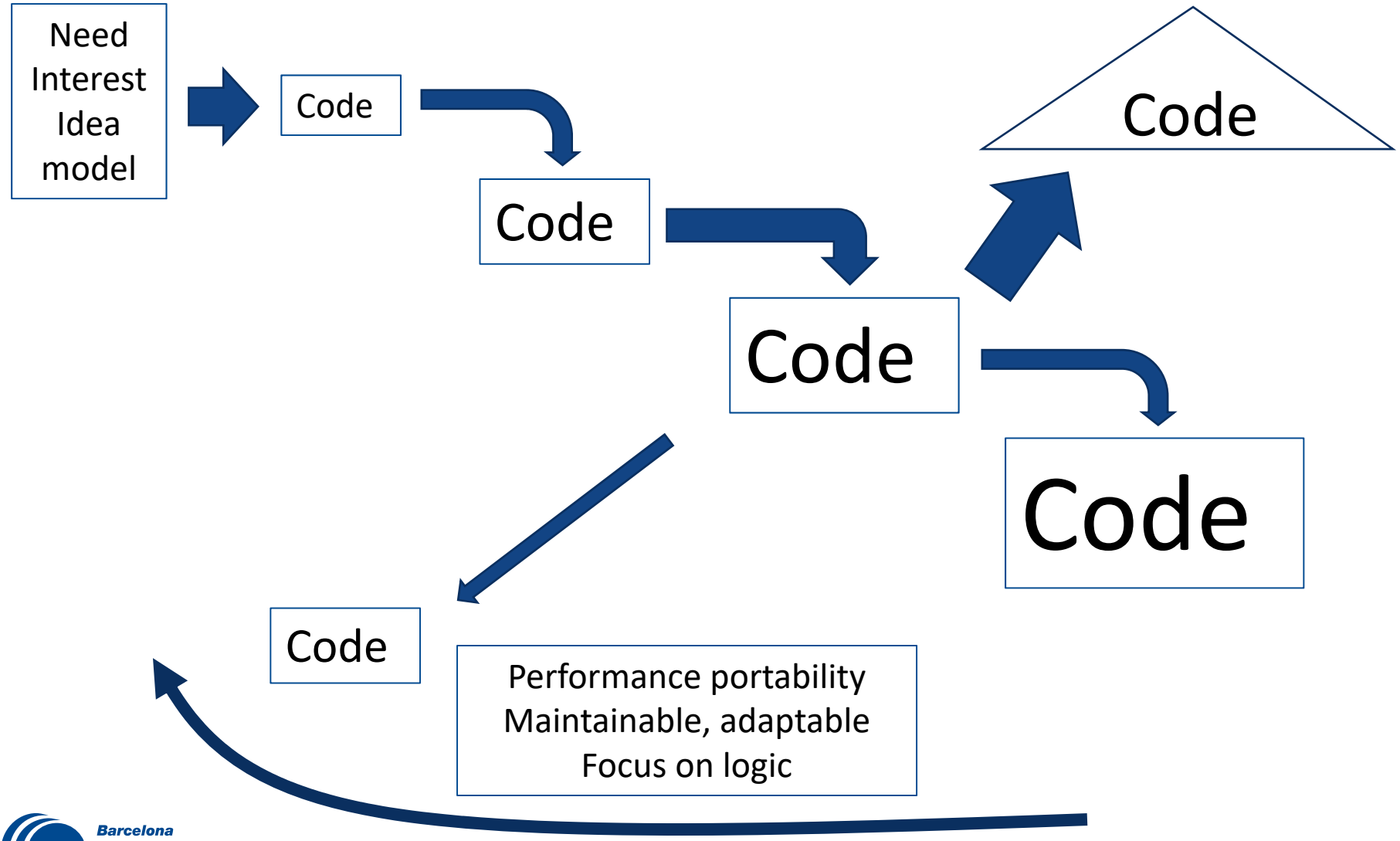
HOPE !!!

Code lifetime cycle



"Platform/language specificities"
"Optimizations"
"Hardwired order/schedules"

There is HOPE !!!



Revolution? Fundamentals?

- An age changing revolution
  - From the latency age …
    - I need something … I need it now !!!
    - Performance dominated by latency in a broad sense
      - At all levels: sequential and parallel
      - Memory and communication, control flow, synchronizations
  - …to the throughput age
    - Ability to instantiate “lots” of work and avoid stalling for specific requests
      - I need this and this and that … and as long as it keeps coming I am ok
      - About tolerating variability by adapting to dynamic situation
    - Performance dominated by overall availability/balance of resources

Key technologies

- Programming models
  - Decouple programmer from machine
  - Enable productive/evolutionary/composable approaches
  - Can we avoid/contain the complexity explosion ?

- Performance analysis tools
  - Avoid flying blind !!
  - Towards insight for application & system developers

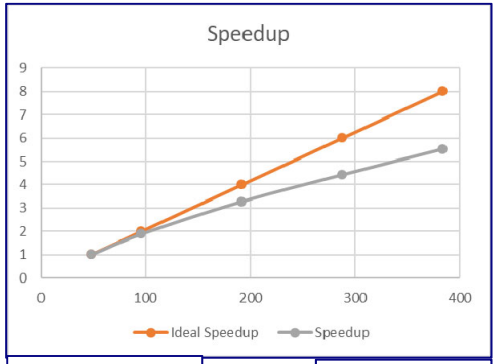
Barcelona Supercomputing Center
Centro Nacional de Supercomputación

EXCELENCIA SEVERO OCHOA

Insight on behavior

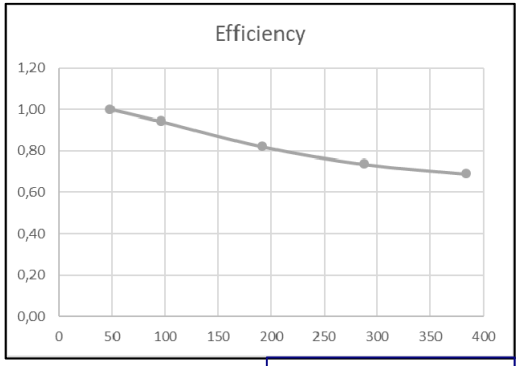
Understand fundamental effects

- Who to blame ?
  - Causes / Effects ?
  - Measurements
    - Too much aggregation ?
    - Too much detail ?



T(48)=11.43 s

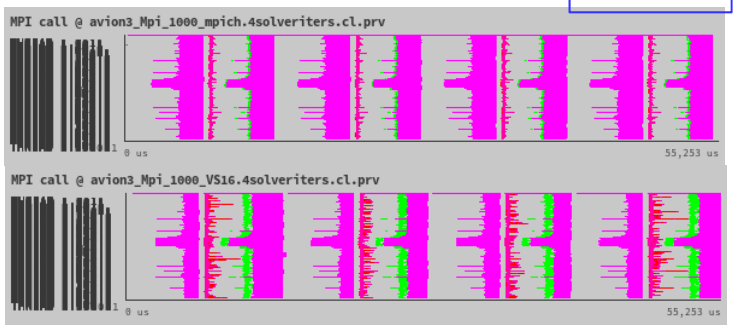
$$Sup = \frac{T_{ref}}{T_P}$$



$$Eff = \frac{Sup}{P / ref}$$

- Understand real issues
  - Fundamentals!
    - Behavior before syntax
  - Don`t mask your symptoms

1k procs



- Basis to fix and counteract

Understanding behaviour

- High level efficiency model
  - Abstract, fundamentals
- Detail
  - Timelines, Histograms
  - What if
  - Performance analytics (clustering, tracking, folding, …)

Performance Optimisation and Productivity Center of Excellence

Providing services

1. Application Performance Audit → Identify performance issues
2. Application Performance Plan → Follow-up analysis
3. Proof-of-Concept → Show effect of proposed optimizations

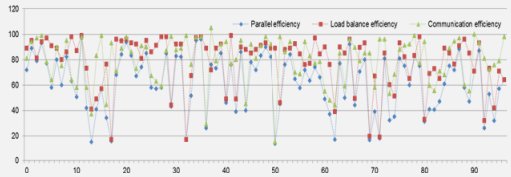
191 services in 30 months

- 169 Assessments + 22 Proof of concept
- 75% of the analyzed codes may benefit from improvements
- 95% customers satisfied with the service
- 50% users requested a second study

Community development

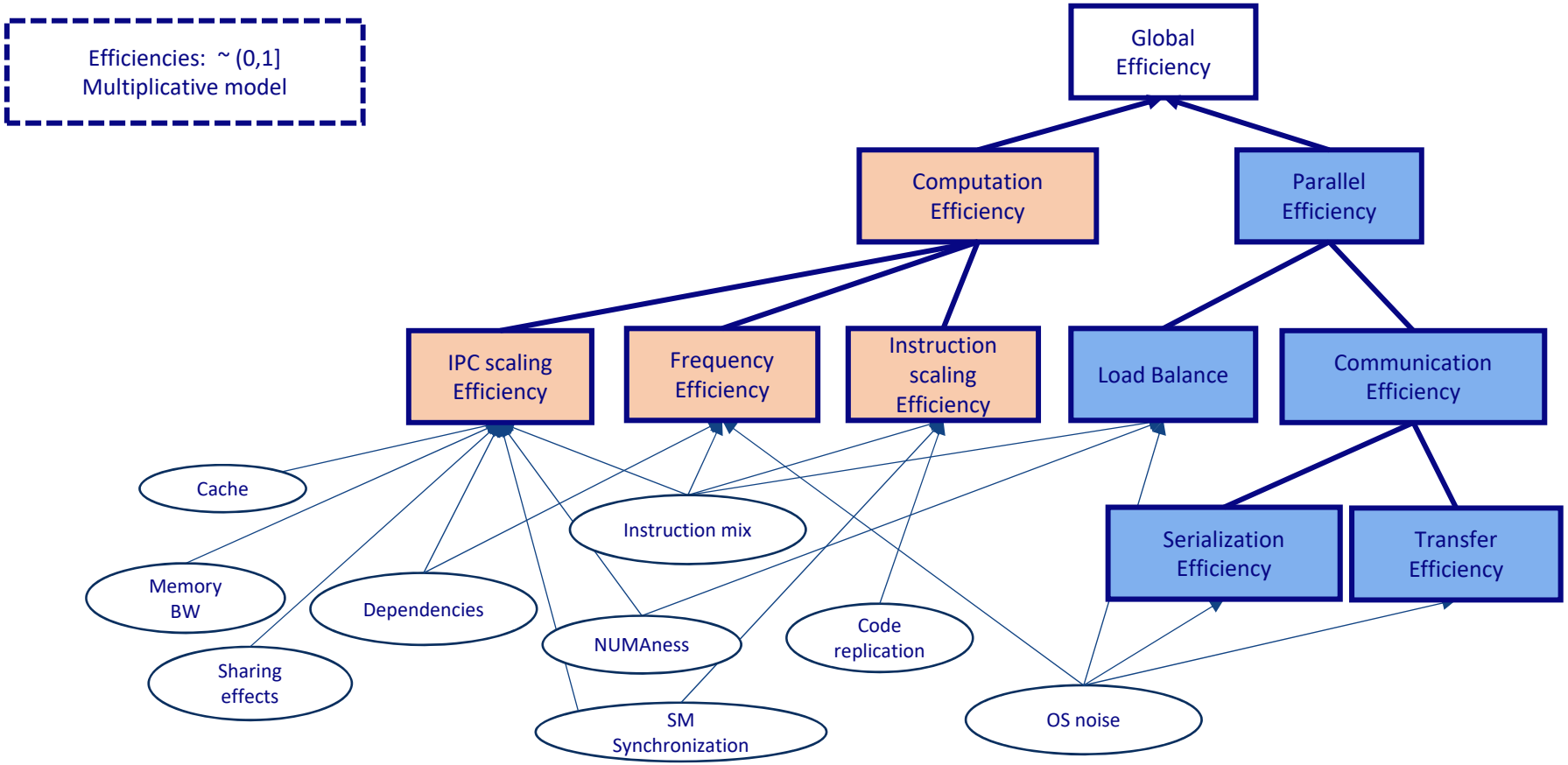
- 1385 potential POP users identified (from 26 EU countries + 3 EFTA countries)
- 149 POP users - 25% from industry
- 50% open source, 25% ISV, 25% in-house

Collecting data characterizing codes

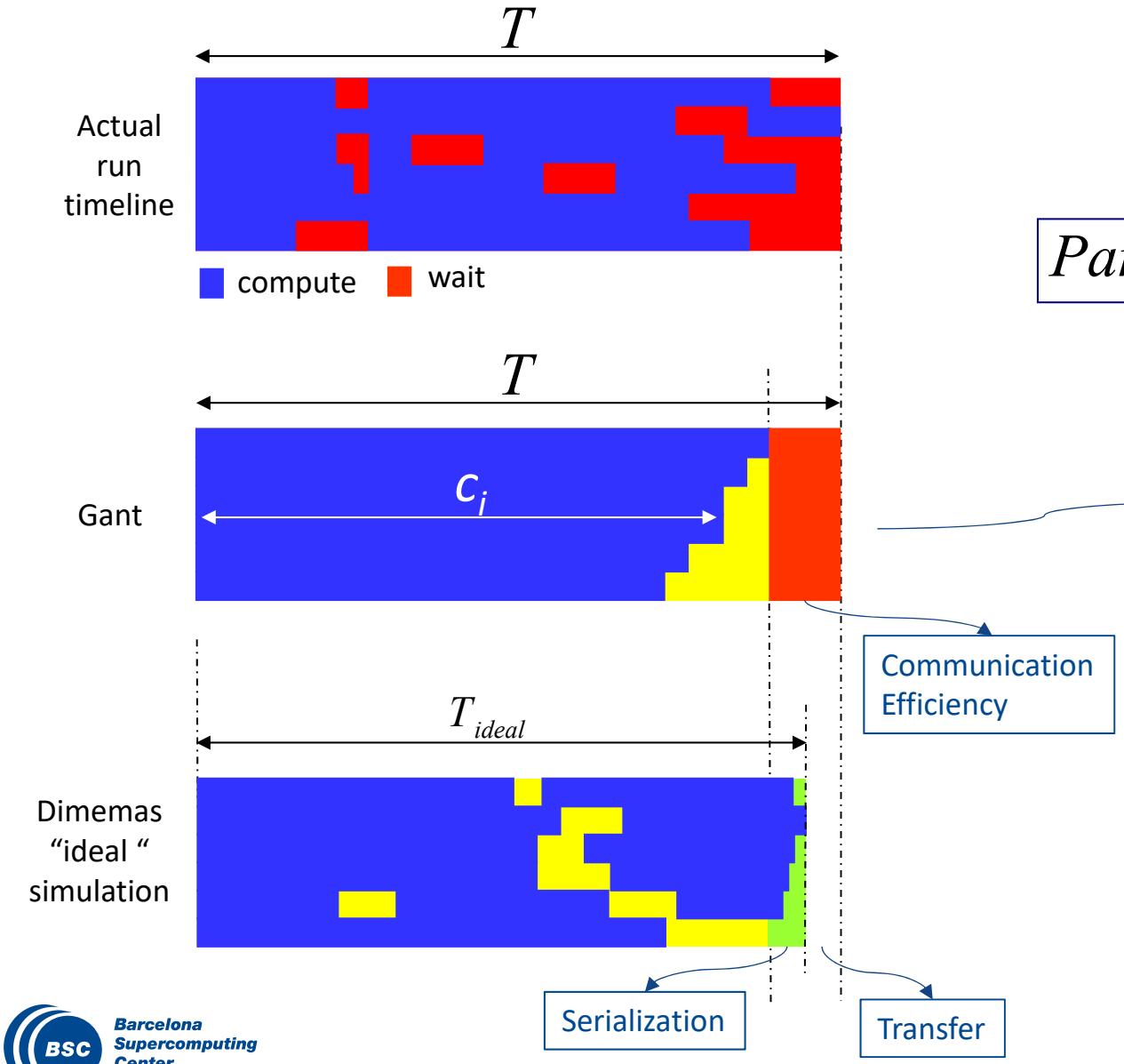


https://www.pop-coe.eu

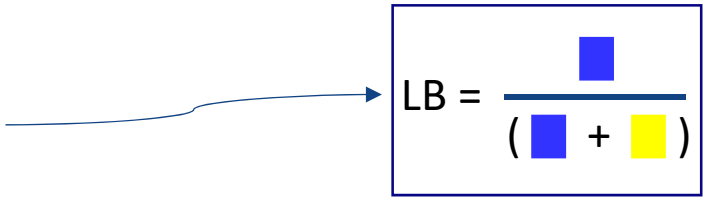
Hierarchical Performance Model …



Parallel Efficiency Model



$$ParEff = LB * Ser * Trf$$



Communication Efficiency

M. Casas et al, "Automatic analysis of speedup of MPI applications". ICS 2008.

Overall behavior awareness …

	2	4	8	16
Parallel Efficiency	0.983	0.944	0.898	0.848
Load Balance	0.987	0.969	0.910	0.918
Serialization efficiency	0.998	0.977	0.994	0.940
Transfer Efficiency	0.998	0.997	0.993	0.983
Computation Efficiency	1.000	0.959	0.868	0.695
IPC scalability	1.000	0.993	0.959	0.842
Instruction scalability	1.000	0.972	0.939	0.908
Frequency scalability	1.000	0.993	0.964	0.910
Global efficiency	0.983	0.905	0.780	0.589

	8	16	32	40
Parallel Efficiency	0.377	0.348	0.222	0.181
Load Balance	0.382	0.360	0.233	0.189
Serialization efficiency	0.981	0.967	0.957	0.959
Transfer Efficiency	1.000	1.000	0.999	0.999
Computation Efficiency	1.000	0.840	0.796	0.774
IPC scalability	1.000	0.944	0.894	0.870
Instruction scalability	1.000	1.000	1.000	0.999
Frequency scalability	1.000	0.890	0.890	0.890
Global efficiency	0.377	0.292	0.177	0.141

	2	4	8
Parallel Efficiency	0.985	0.914	0.931
Load Balance	0.985	0.914	0.939
Serialization efficiency	1.000	1.000	0.911
Transfer Efficiency	1.000	1.000	1.088
Computation Efficiency	1.000	0.814	0.633
IPC scalability	1.000	0.961	0.594
Instruction scalability	1.000	0.873	1.106
Frequency scalability	1.000	0.970	0.964
Global efficiency	0.985	0.744	0.590

	32	48	64	96	128	256
Parallel Efficiency	0.917	0.906	0.887	0.847	0.864	0.790
Load Balance	0.946	0.925	0.934	0.858	0.871	0.813
Serialization efficiency	0.970	0.980	0.951	0.987	0.994	0.976
Transfer Efficiency	1.000	1.000	1.000	0.999	0.999	0.995
Computation Efficiency	1.000	1.025	1.026	1.036	1.012	0.956
IPC scalability	1.000	1.013	1.013	1.013	1.004	0.982
Instruction scalability	1.000	1.013	1.020	1.019	1.009	0.977
Frequency scalability						
Global efficiency	0.917	0.928	0.911	0.877	0.874	0.755

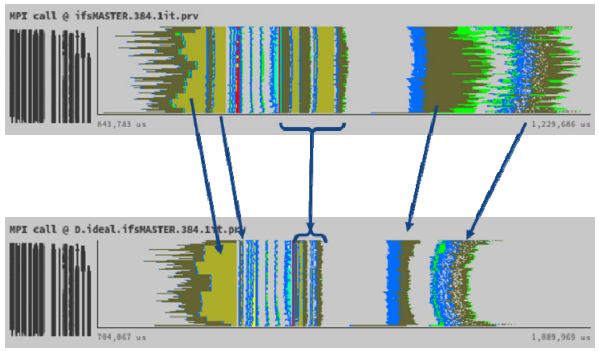
	48	96	192	288	384
Parallel Efficiency	0.865	0.843	0.760	0.744	0.707
Load Balance	0.917	0.900	0.904	0.880	0.896
Serialization efficiency	0.975	0.989	0.972	0.963	0.956
Transfer Efficiency	0.967	0.948	0.866	0.878	0.826
Computation Efficiency	1.000	0.966	0.932	0.856	0.843
IPC scalability	1.000	0.974	0.955	0.896	0.891
Instruction scalability	1.000	0.993	0.976	0.950	0.943
Frequency scalability	1.000	0.999	1.000	1.006	1.003
Global efficiency	0.865	0.815	0.709	0.637	0.596

Coloring	1.000	0.850	0.849	0.750	0.749	0.650	0.649	0.000

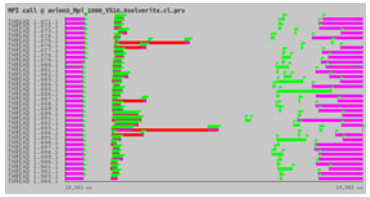
… and detail

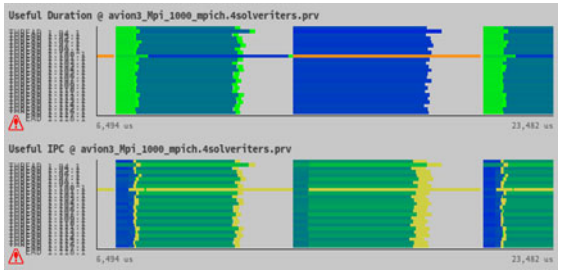
- Macroscopic properties depends on microscopic interactions

What if MPI had no overhead and transfer was instantaneous ?

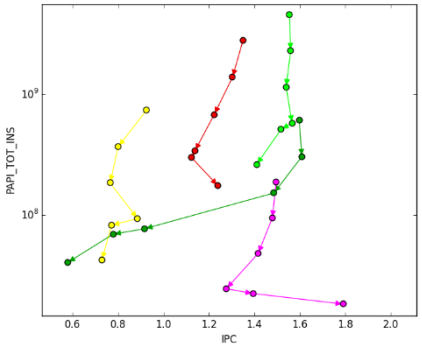


Fundamental underlying causes?





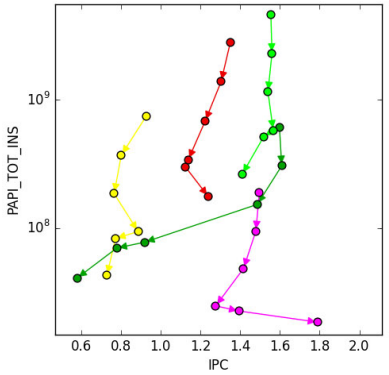
How microscopic metrics change with scale ?

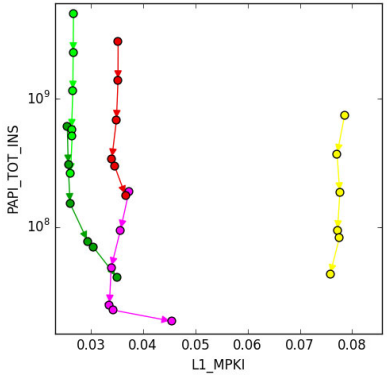


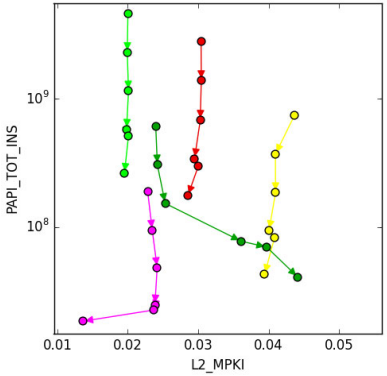
- … that have to be well understood to take informed counter measures

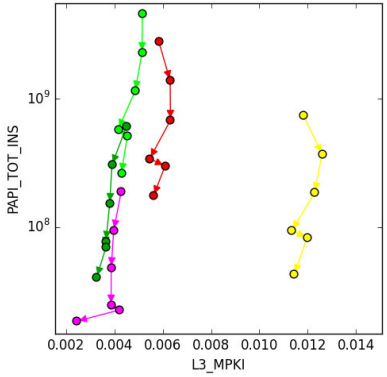
… and detail

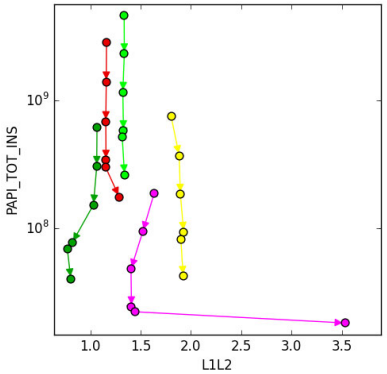
- Tracking IPC factors with scale

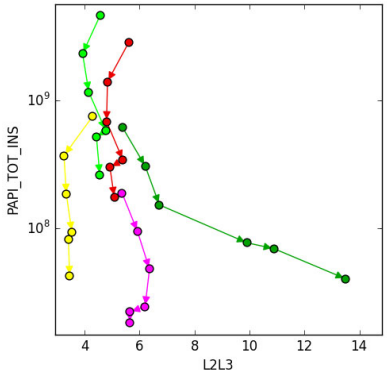


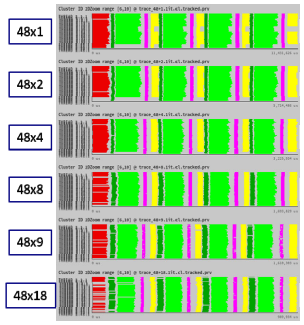








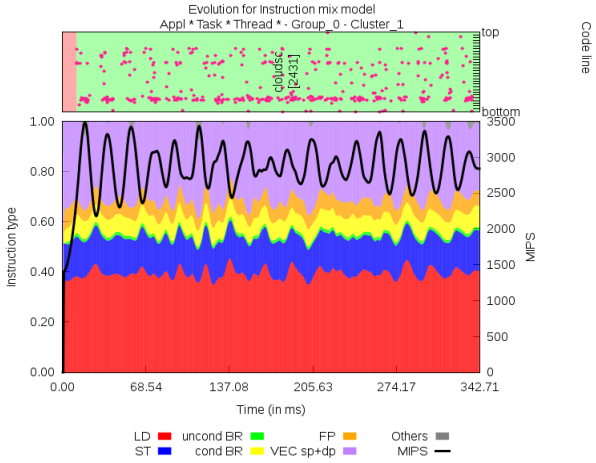




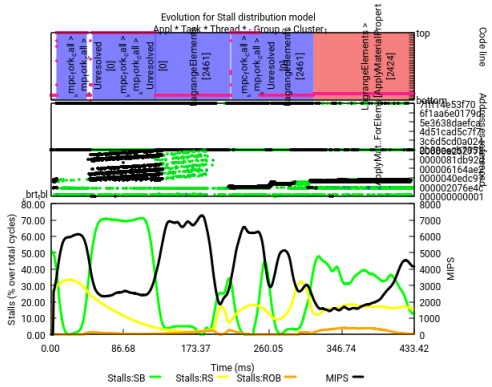
… and detail

- Instantaneous metrics and data access pattern

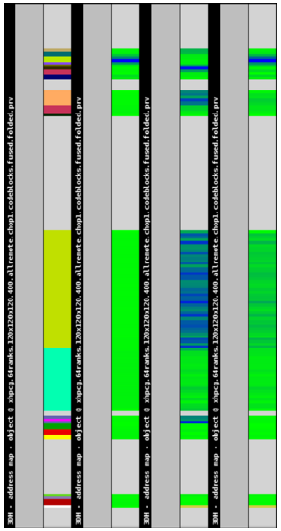
Instantaneous metrics for ALL hardware counters at “no” cost

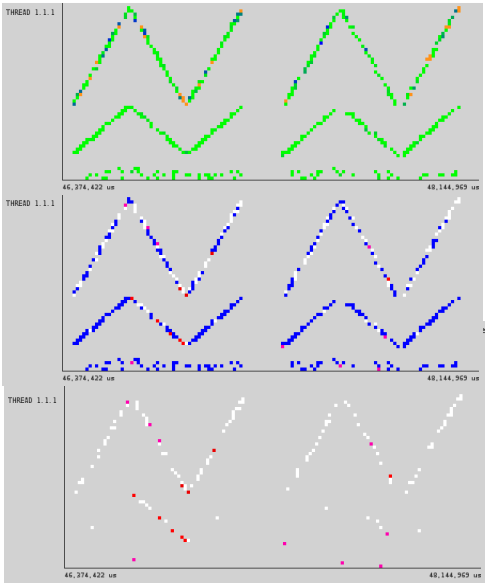


BSC-ES – EC-EARTH



Lulesh 27 procs
rank 23





Data access patterns

HPCG 64 procs
rank 0

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

EXCELENCIA SEVERO OCHOA

Programming models

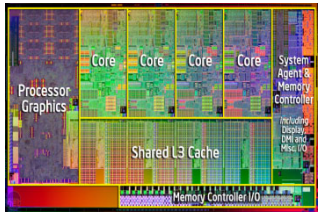
Vision in the programming revolution

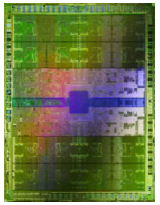
Applications

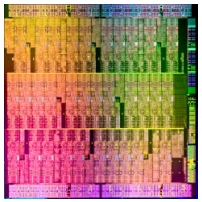
PM: High-level, clean, abstract interface

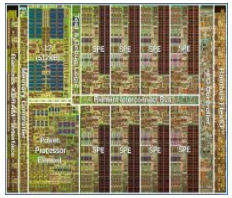
Power to the runtime

ISA / API









General purpose

Decouple

Forget about resources

Minimal & sufficient permeability?

Intelligence
&
Resource management

“Reuse & expand” old architectural ideas under new constraints

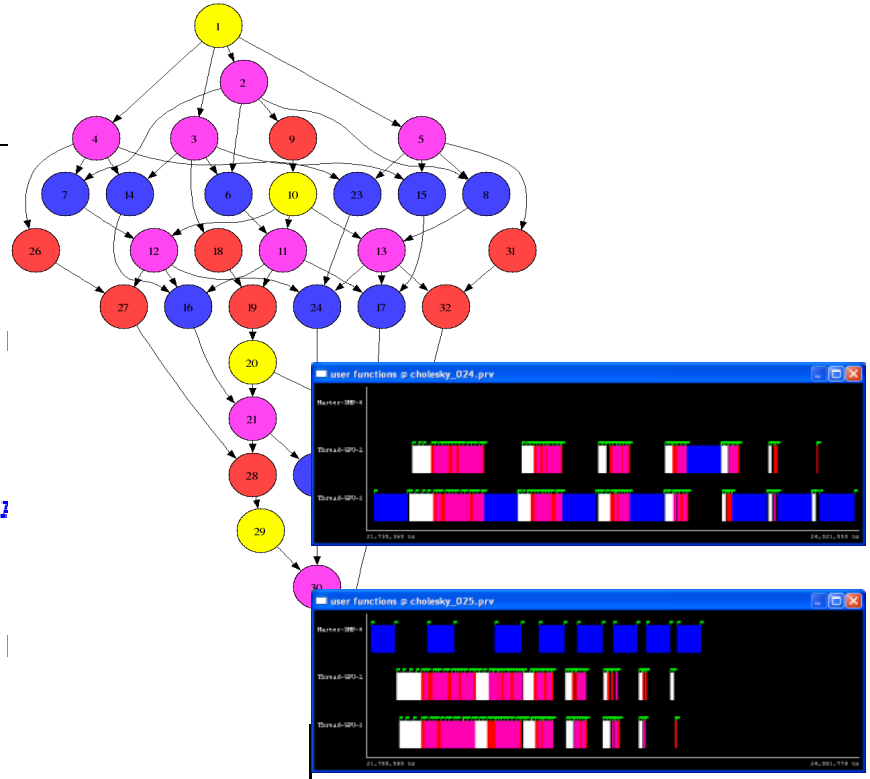
The StarSs family of programming models

- **A sequential task based** program …
- … on a **single address/name space …**
- **… with directionality annotations …**
- …that happens to **execute in parallel**

Integrate concurrency and data

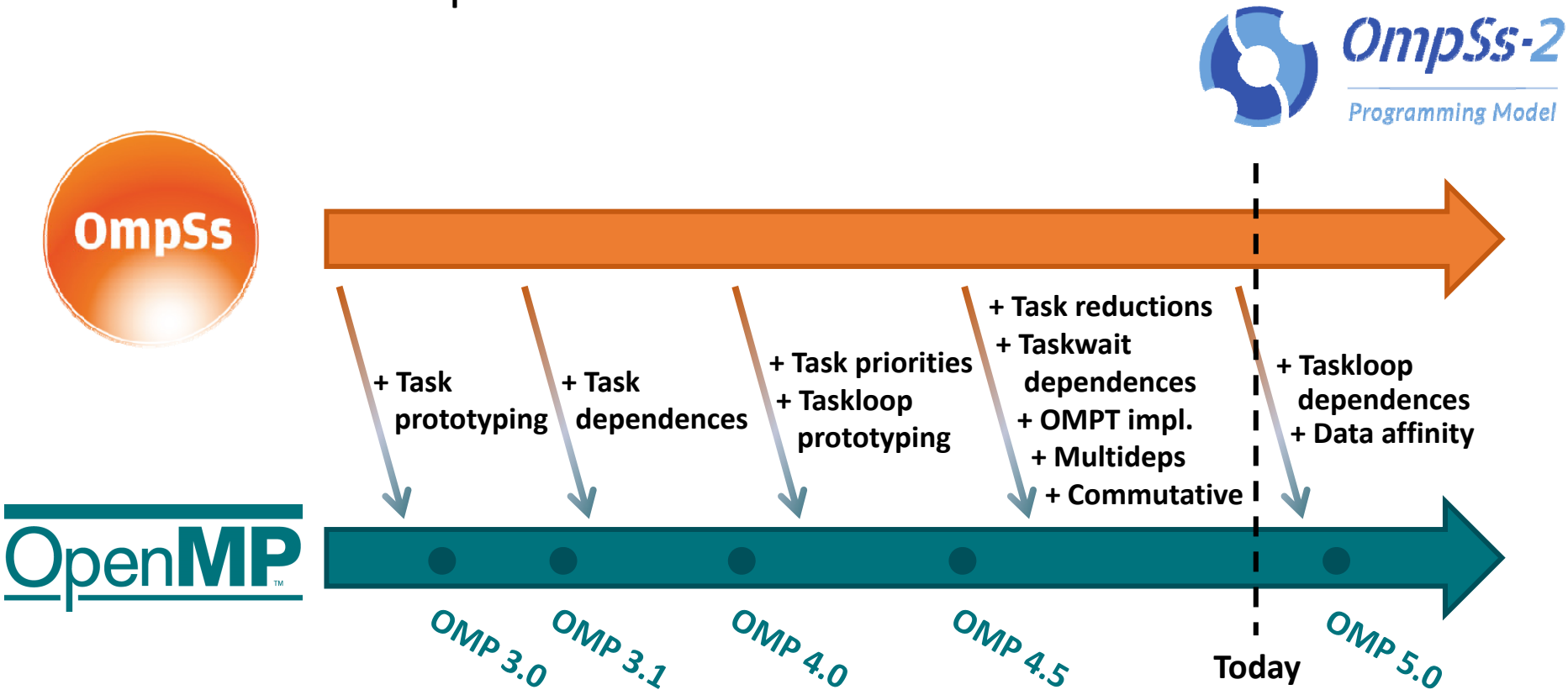
- Single mechanism
  - Concurrency:
    - Dependences built from data accesses
    - Lookahead: About instantiating work
  - Locality & data management
    - From data accesses

```
void Cholesky(int NT, float *A[NT][NT] ) {
   for (int k=0; k<NT; k++) {
      #pragma omp task inout ([TS][TS](A[k][k]))
      spotrf (A[k][k], TS) ;
      for (int i=k+1; i<NT; i++) {
        #pragma omp task in([TS][TS](A[k][k])) inout ([TS]
        strsm (A[k][k], A[k][i], TS);
      }
      for (int i=k+1; i<NT; i++) {
        for (j=k+1; j<i; j++) {
          #pragma omp task in([TS][TS](A[k][i]), [TS][TS](
                           inout ([TS][TS](*A[j][i]))
          sgemm( A[k][i], A[k][j], A[j][i], TS);
        }
        #pragma omp task in ([TS][TS](A[k][i])) inout([TS]
        ssyrk (A[k][i], A[i][i], TS);
      }
   }
}
```



OmpSs

- A forerunner for OpenMP

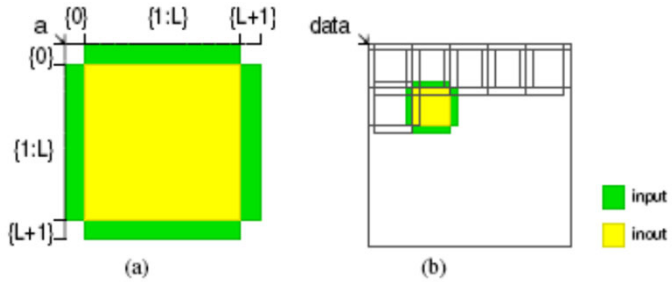


Some important topics/developments

- Regions
- **Nesting**
- Taskloops + dependences
- Hints
- **MPI Interoperability**
- **Malleability**
- Homogenize Heterogeneity
- Hierarchical “acceleration”
- Memory management & Locality*

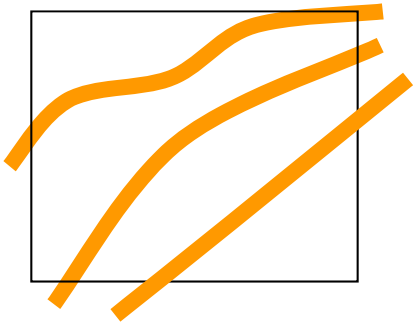
Regions

- Precise nD subarray accesses
  - "Complex" analysis but …
- Enabler for …
  - Recursion
  - Flexible nesting
  - Taskloop dependences
  - Data management
    - locality
    - layout



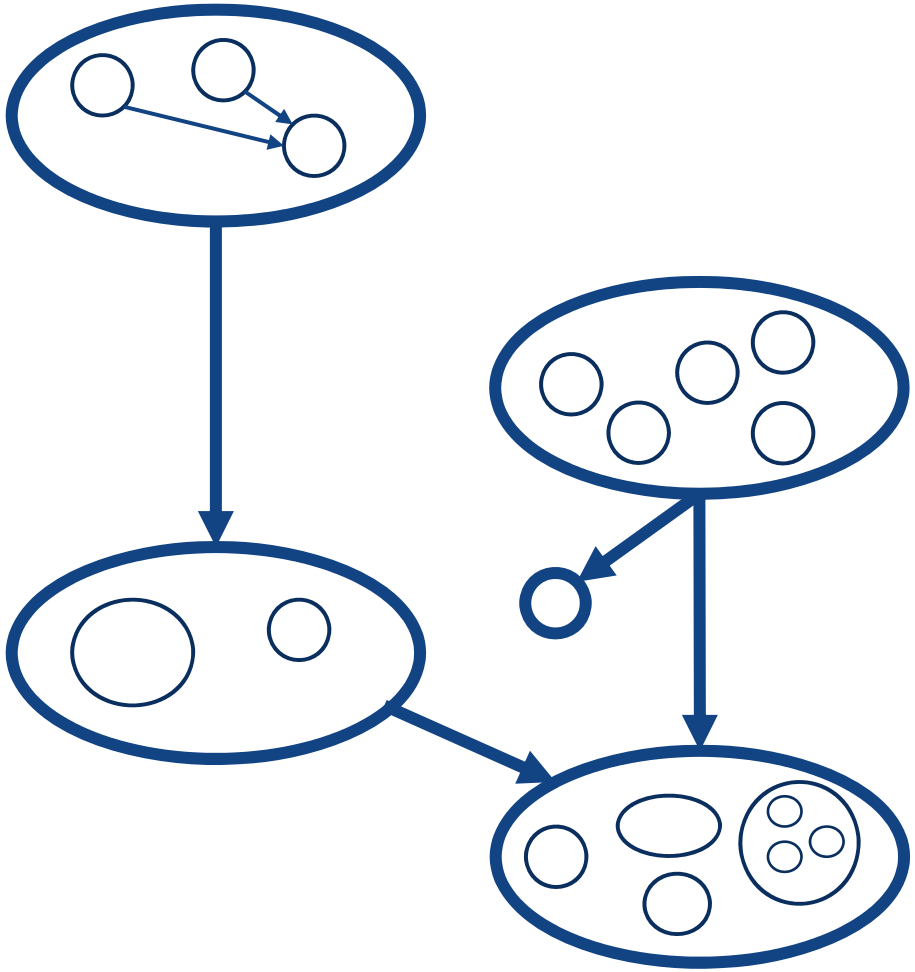

```
void gs (float A[(NB+2)*BS][(NB+2)*BS])
{
  int it,i,j;

  for (it=0; it<NITERS; it++)
    for (i=0; i<N-2; i+=BS)
      for (j=0; j<N-2; j+=BS)
        gs_tile(&A[i][j]);
}
```

```
#pragma omp task \
  in(A[0][1;BS], A[BS+1][1;BS], \
       A[1;BS][0], A[1:BS][BS+1]) \
  inout(A[1;BS][1;BS])
void gs_tile (float A[N][N])
{
  for (int i=1; i <= BS; i++)
    for (int j=1; j <= BS; j++)
      A[i][j] = 0.2*(A[i][j] + A[i-1][j] +
                A[i+1][j] + A[i][j-1] +
                A[i][j+1]);
}
```



Nesting

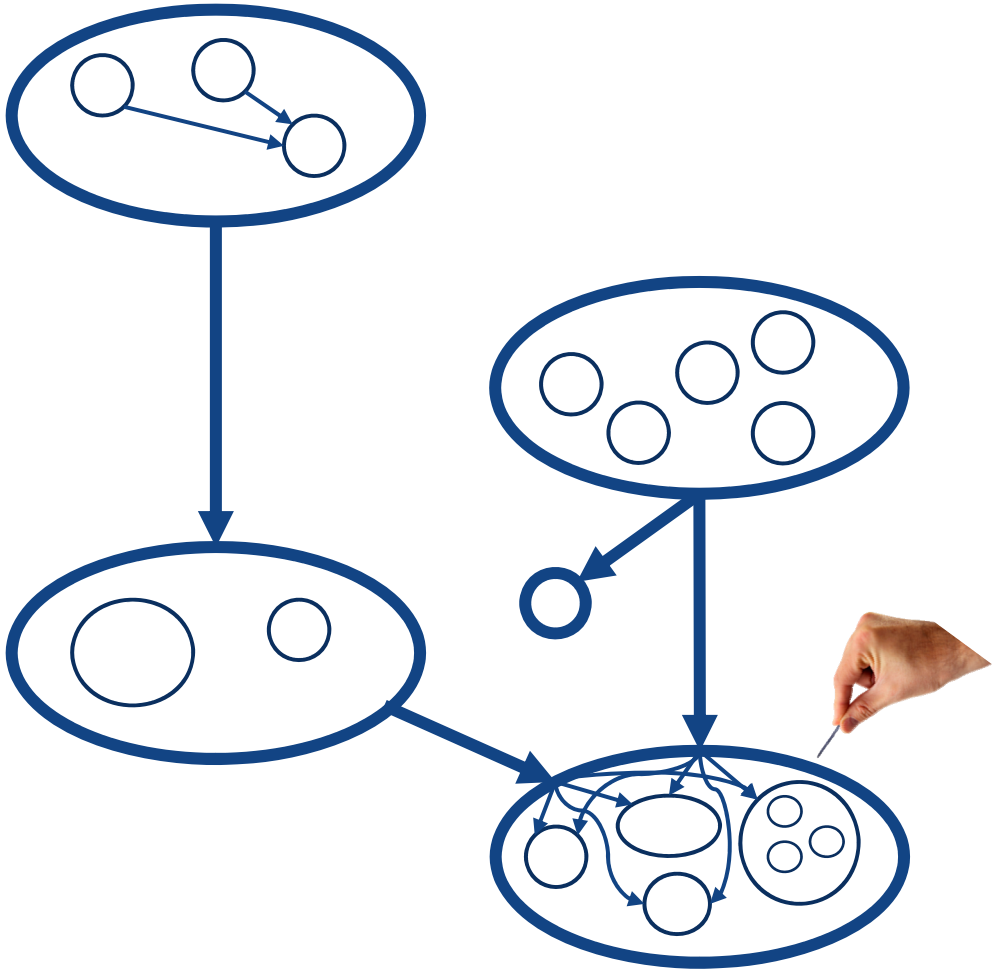
- Top down
  - Every level contributes
- Flattening dependence graph
  - Increase concurrency
  - Take out runtime overhead from critical path
- Granularity control
  - final clauses, runtime



J. M. Perez, et all, "Improving the Integration of Task Nesting and Dependencies in OpenMP" IDPS 2017

Nesting

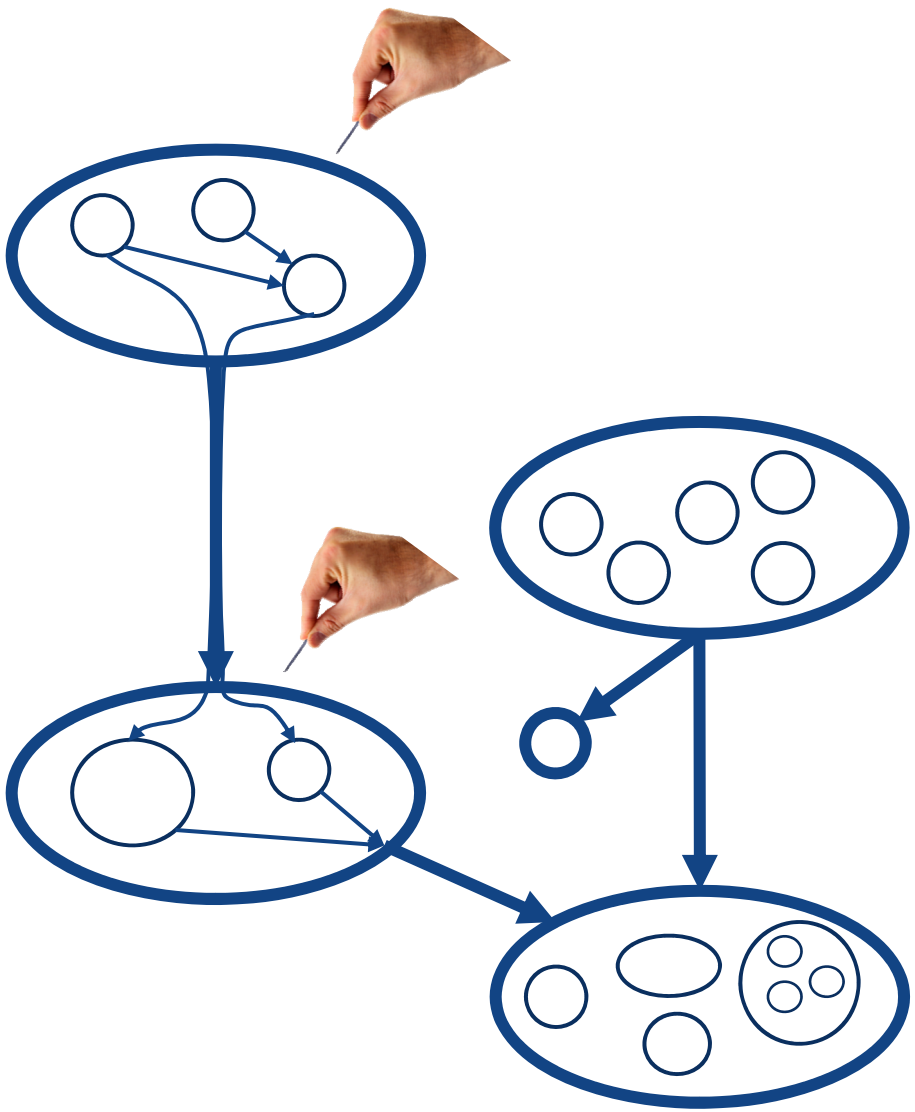
- Top down
  - Every level contributes
- Flattening dependence graph
  - Increase concurrency
  - Take out runtime overhead from critical path
- Granularity control
  - final clauses, runtime



J. M. Perez, et all, "Improving the Integration of Task Nesting and Dependencies in OpenMP" IPDPS 2017

Nesting

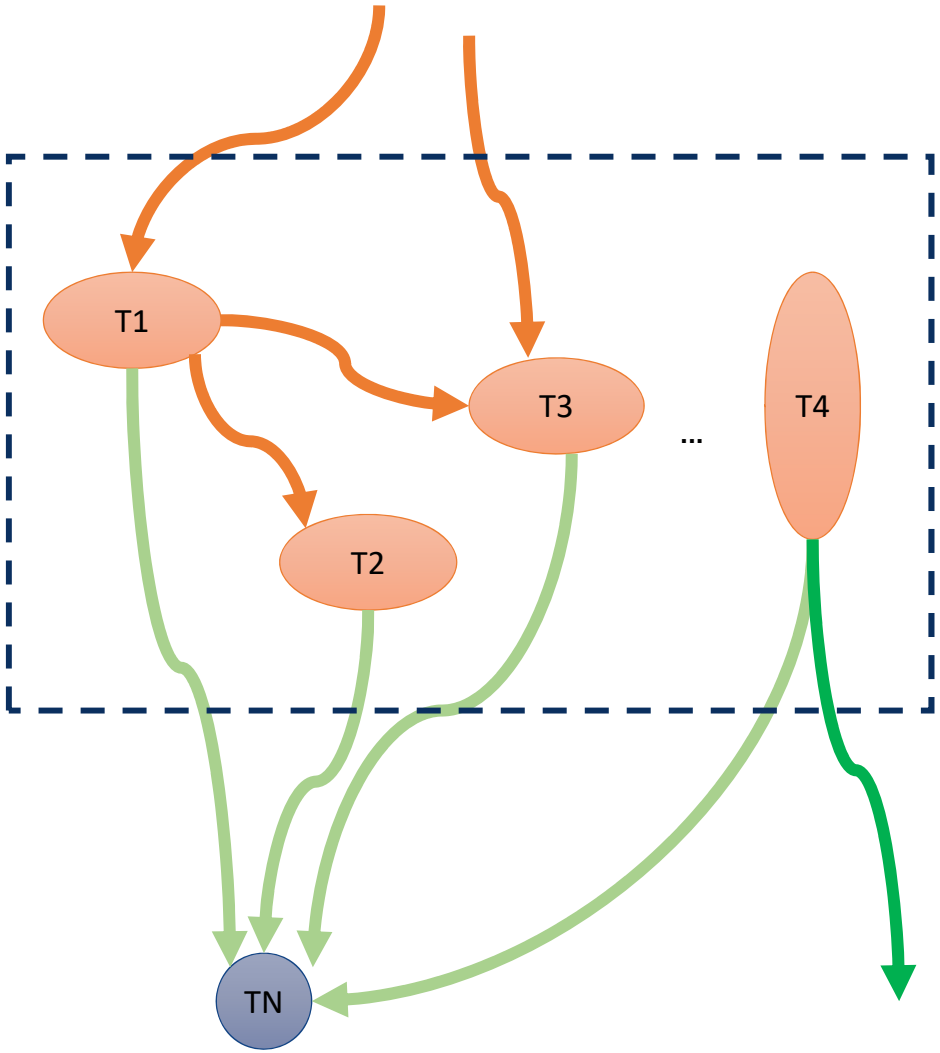
- Top down
  - Every level contributes
- Flattening dependence graph
  - Increase concurrency
  - Take out runtime overhead from critical path
- Granularity control
  - final clauses, runtime



J. M. Perez, et all, "Improving the Integration of Task Nesting and Dependencies in OpenMP" IDPS 2017

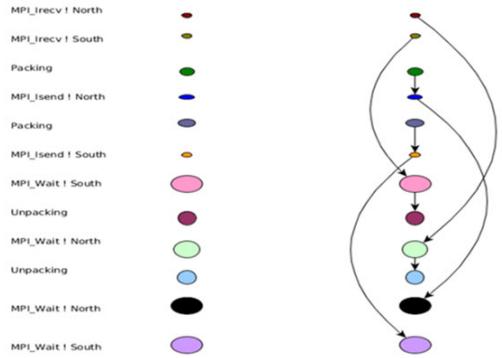
Taskloops & dependences

- Dependences
  - Intra loop
  - Inter loops
- Dynamic granularities
  - Guided
  - Runtime
- Combination
  - Enabled by regions support



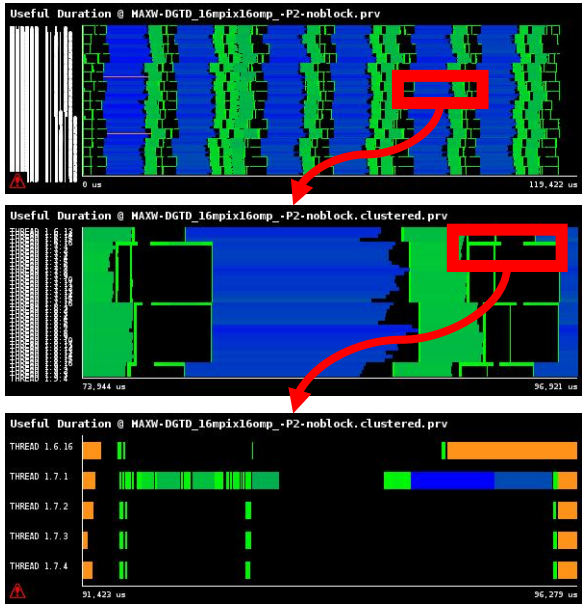
MPI - OpenMP interoperability

- Hybrid Amdahl's law
  - A fairly "bad message" for programmers
- Significant non parallelized parts
  - pack/unpack
    - Often too fine grain
    - Significant variability
  - MPI calls
    - Too serial
      - Communicator context
      - MPI order semantics
        - Instead of tags
      - Hardwired schedules



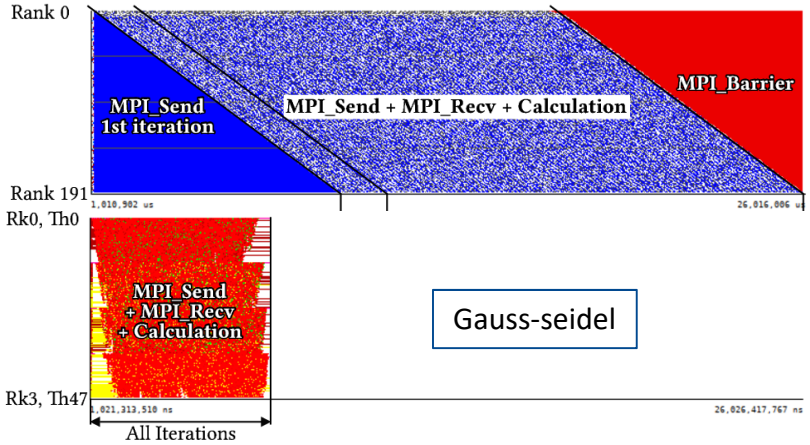

```
for ()
   pack
   irecv
   isend
wait all sends
for ()
   test
   unpack
```

MAXW-DGTD

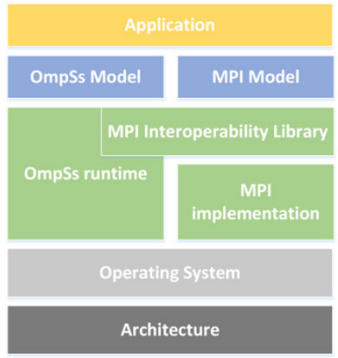


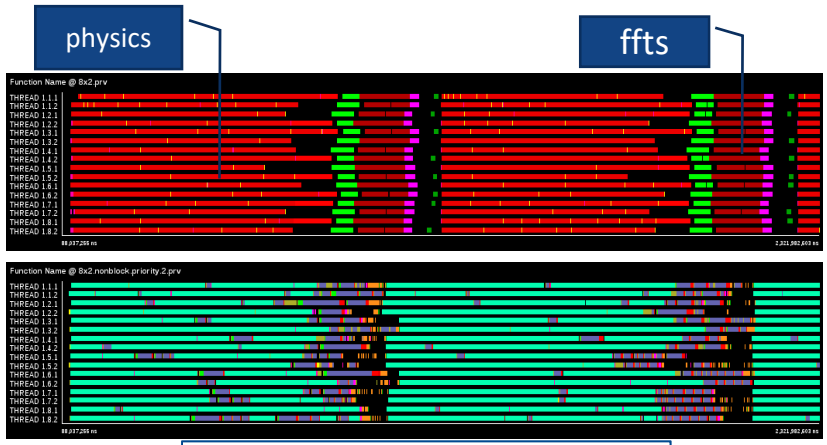
MPI - OpenMP interoperability

- Taskifying MPI calls
  - Virtualize “communication resource”
- Opportunities
  - Overlap/out of order execution
    - Computation - communication
    - Communication - communication
    - Phases / iterations
  - Provide laxity for communications
    - Tolerate poorer communication
  - Migrate/aggregate load balance issues
    - Flexibility for DLB



Gauss-seidel





IFS weather code kernel. ECMWF

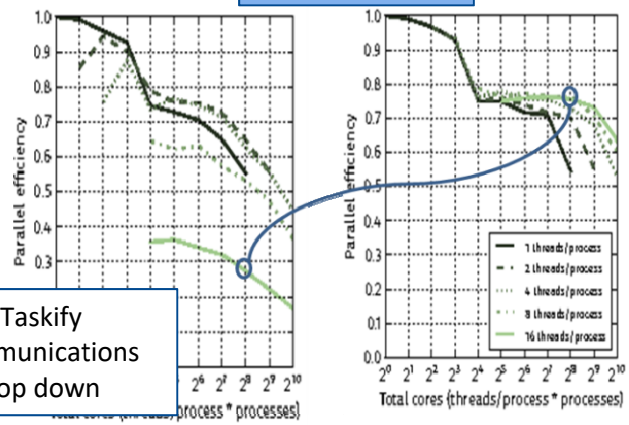
V. Marjanovic et al, “Overlapping Communication and Computation by using a Hybrid MPI/SMPSs Approach” ICS 2010

K. Sala et al, "Extending TAMPI to support asynch MPI primitives”. OpenMPCon18

Towards throughput oriented hybrid programming

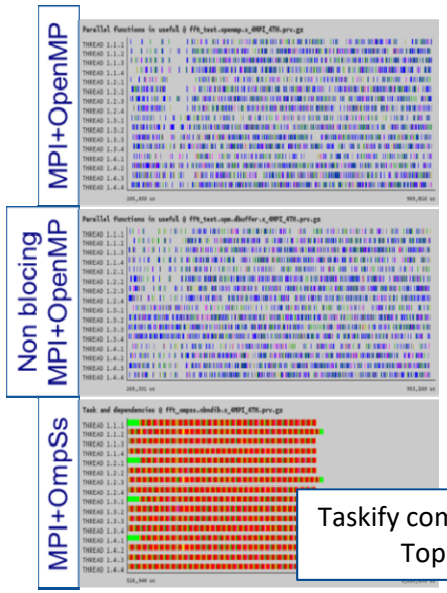
- Programmer Attitude
- Top down
- Try to increase granularity
- Ease of restructuring
- Recognizable code

NT-CHEM

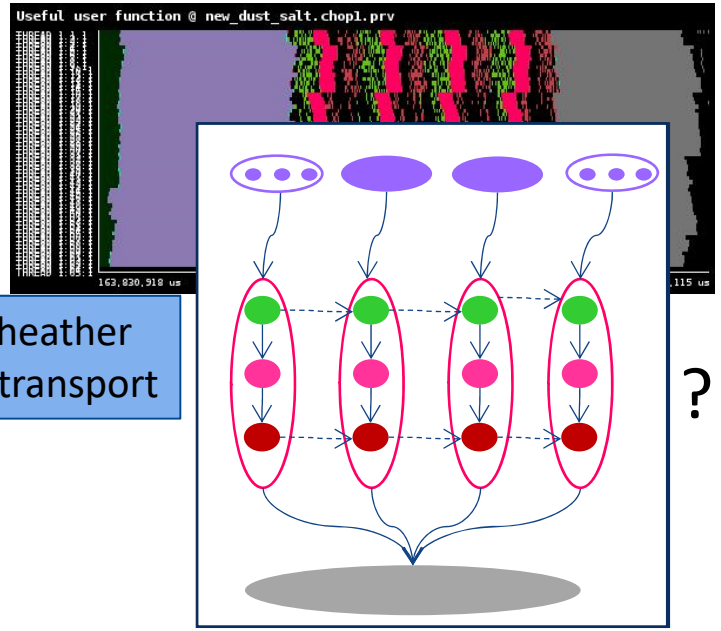


Taskify communications Top down

FFTlib (QE miniapp)



Taskify communications Top down



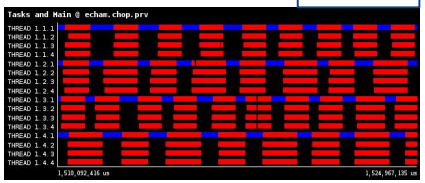
NMMB: wheather + chemical transport

?

Exploiting malleability

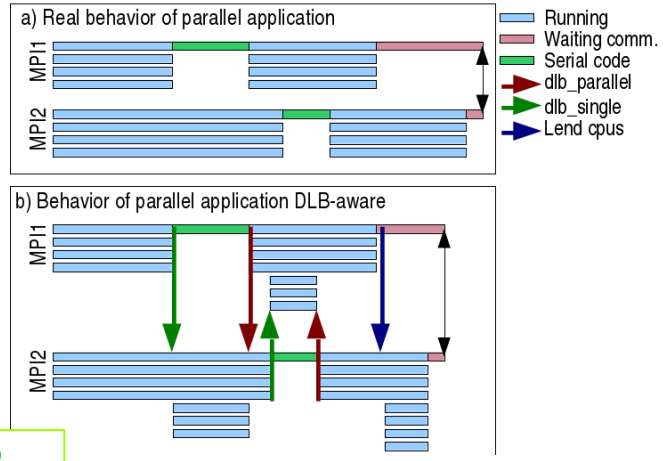
- Malleability
  - ~~Omp_get_thread_num~~, ~~Thread private~~, ~~large parallels~~ ….
- Dynamic Load Balance & Resource management
  - Intra/inter process/application
- Library (DLB)
  - Runtime interception (MPIP, OMPT, …)
  - API to hint resource demands
  - Core reallocation policy
- Opportunity to fight Amdalh's law
  - Productive / Easy !!!
    - Hybridize only imbalanced regions
    - Nx1

ECHAM



dynamic load balancing

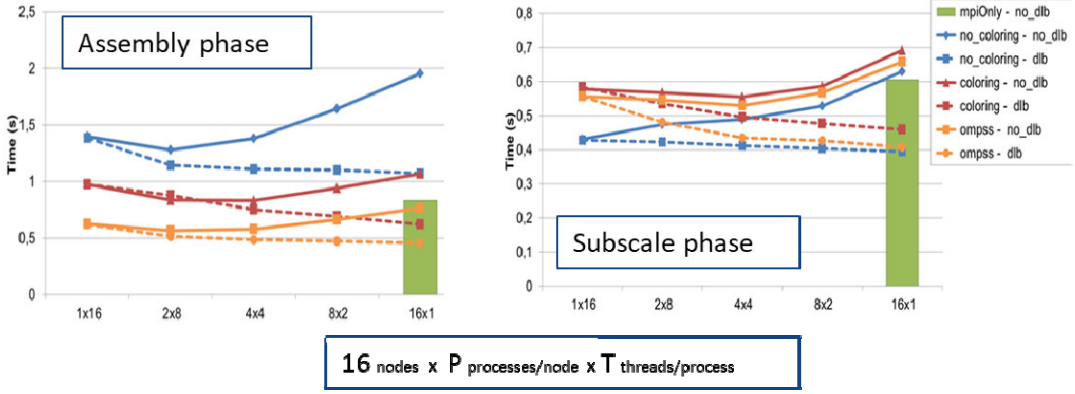
https://pm.bsc.es/dlb



"LeWI: A Runtime Balancing Algorithm for Nested Parallelism". M.Garcia et al. ICPP09
"Hints to improve automatic load balancing with LeWI for hybrid applications" JPDC2014

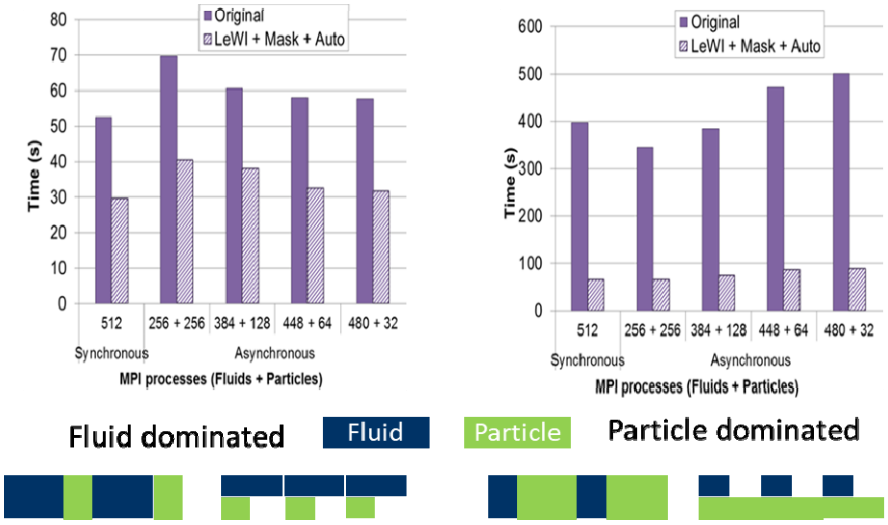
Towards the throughput age

- By
  - Express potential concurrency
  - Malleability
  - Dynamic resource sharing/management



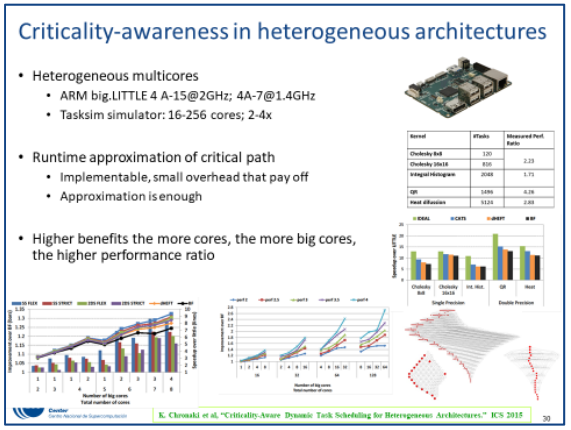
- Configuration independence
  - Amount of resources is what really matters

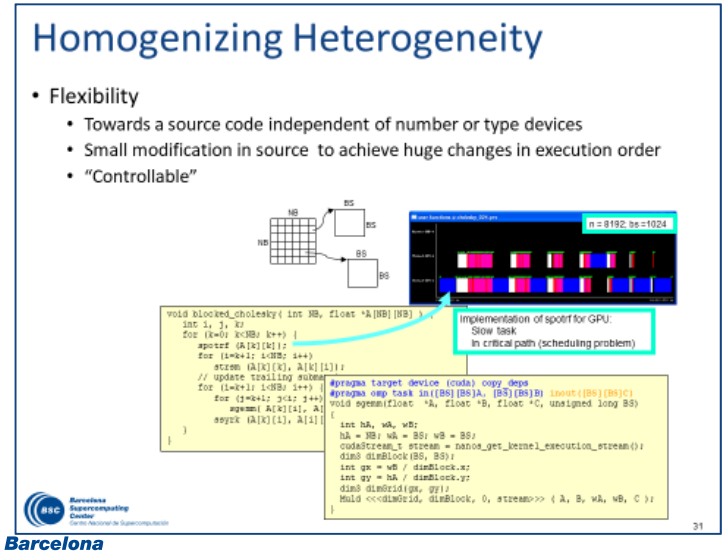
- Side effects
  - Nx1 can be better than pure MPI !!!
  - hope for lazy programmers

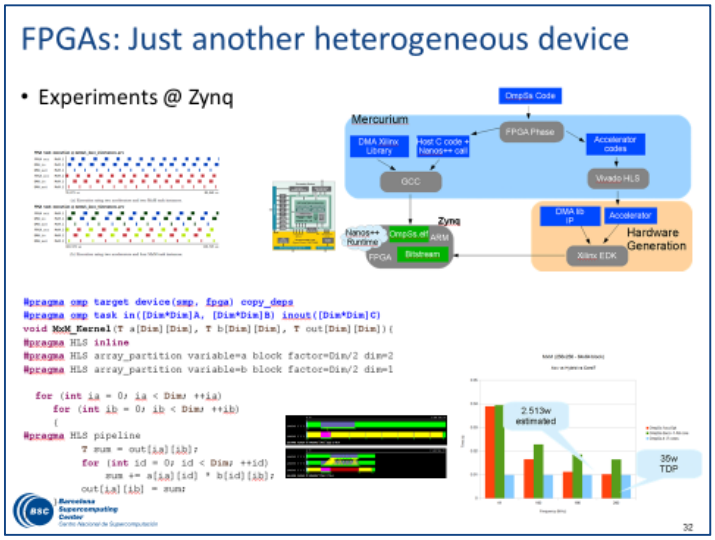


Homogenizing Heterogeneity

- Performance heterogeneity
- ISA heterogeneity
- Several non coherent address spaces







Other languages & granularities

- Task based computational workflows
- PyCOMPSs
  - Decorators
  - Directionality
  - constraints
  - Synchronization API

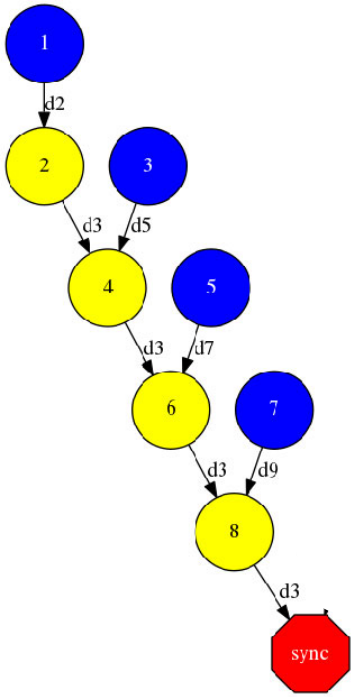
```
Data = [block1, block2, …, blockN]
result=defaultdict(int)
for block in Data:
    presult = word_count(block)
    reduce_count(result, presult)
postprocess(result)
…
```

- COMPSs
  - Java

```
@constraint(ProcessorCoreCount=8)
@task(returns=dict)
def word_count(collection):
    …
```

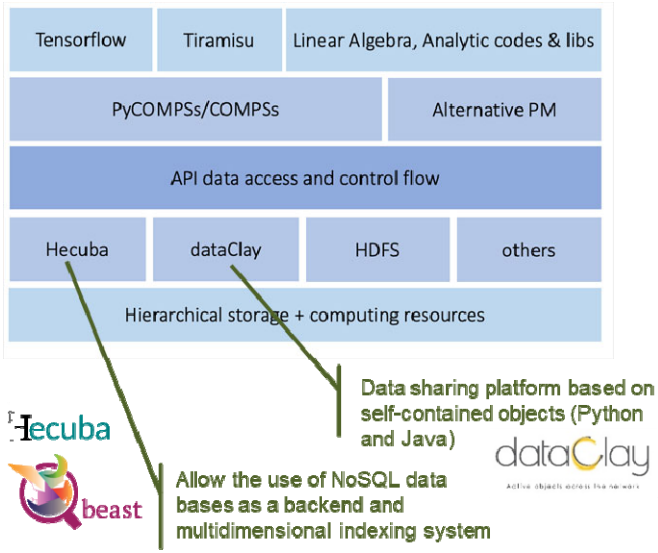
```
@task(dict_1=INOUT)
def reduce_count(dict_1, dict_2):
    …
```

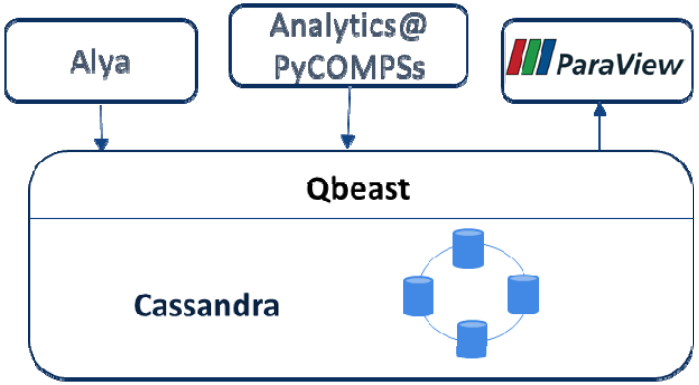
```
@task
def postprocess(dict):
    …
```

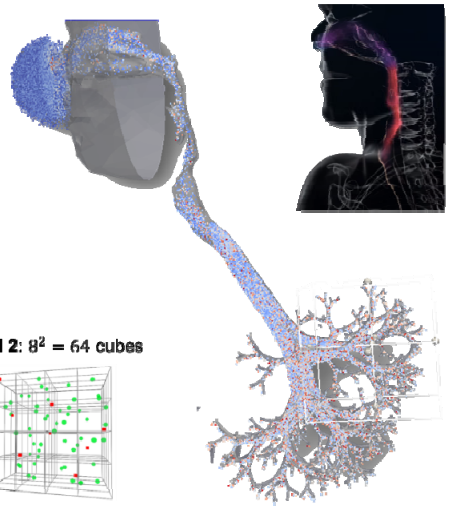


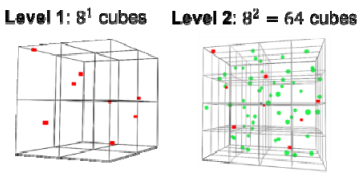
Towards an HPC/HPDA environment

- Integrated Concurrency and Data
  - Python with annotations
  - Persistent object
  - Distributed data and execution
- Use practices
  - On line analytics in different application domains



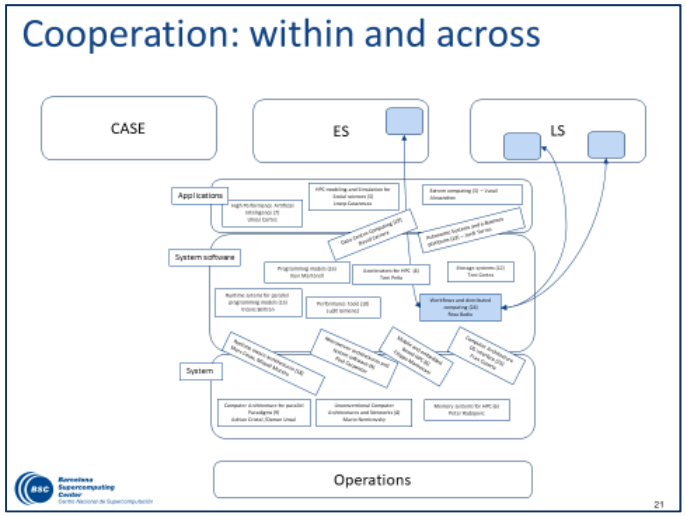


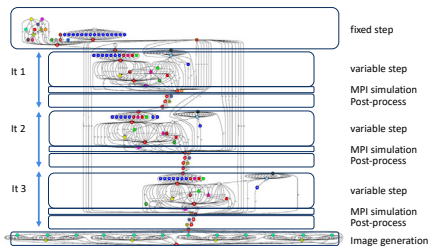


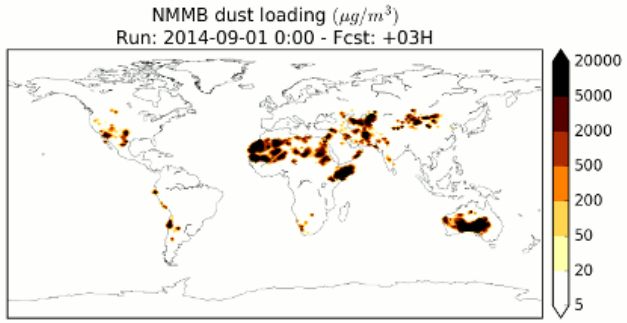


Apps on PyCOMPSs

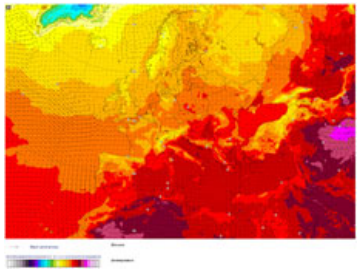
- Used in production runs!!
  - Simulation + analytics workflows
  - Hundreds of nodes, Ks processors
- Co-design insight



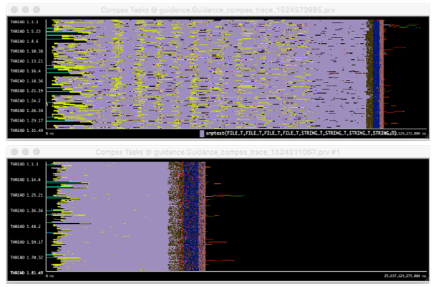


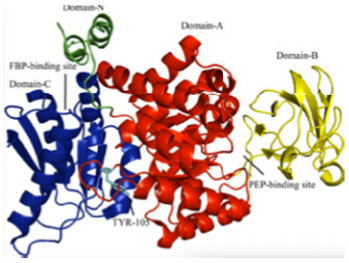


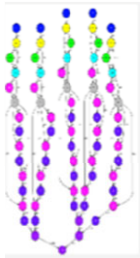
NMMB-Monarch (Weather and dust forecast)



GUIDANCE







PyMDsetup

MUG

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

EXCELENCIA SEVERO OCHOA

Concluding remarks …

Age before beauty

- Behavior (insight/models) before syntax
- Detail performance analytics before aggregated profiles
- Work instantiation and order before overhead
- Malleability before fitted rigid structure
- Possibilities before how tos
- Elegance before one day shine

El abuelo cebolleta
ataca de nuevo

Need to promote best practices !!!

- Task based
  - OmpSs, OpenMP ~ OK, but …
  - "Proper" model does not guarantee "proper" programs
    - Flexibility → can be used "wrong"
    - Legacy features have a strong "shadow"

- Best practices / Methodology
  - Forget about resources
  - Top down
    - Every level contributes
  - Throughput oriented
    - try not to stall !
  - Think global
    - of potentials
    - may be unprecise
  - Specify local
    - needs and outcomes of the functionality being written
    - Precise

The real parallel programming revolution …

- … is in the mindset of programmers

- From latency to throughput oriented !!!

To exascale … and before

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

EXCELENCIA SEVERO OCHOA

Thanks