POP CoE: Understanding applications and how to prepare for exascale

Jesus Labarta (BSC)

EU H2020 Center of Excellence (CoE)

Lecce, May 17th 2018

5th ENES HPC workshop

POP objective

- Promote methodologies and best practices in
  - Performance analysis
  - Parallel programming practices

- By means of services
  - Performance assessments
  - Proof of concept

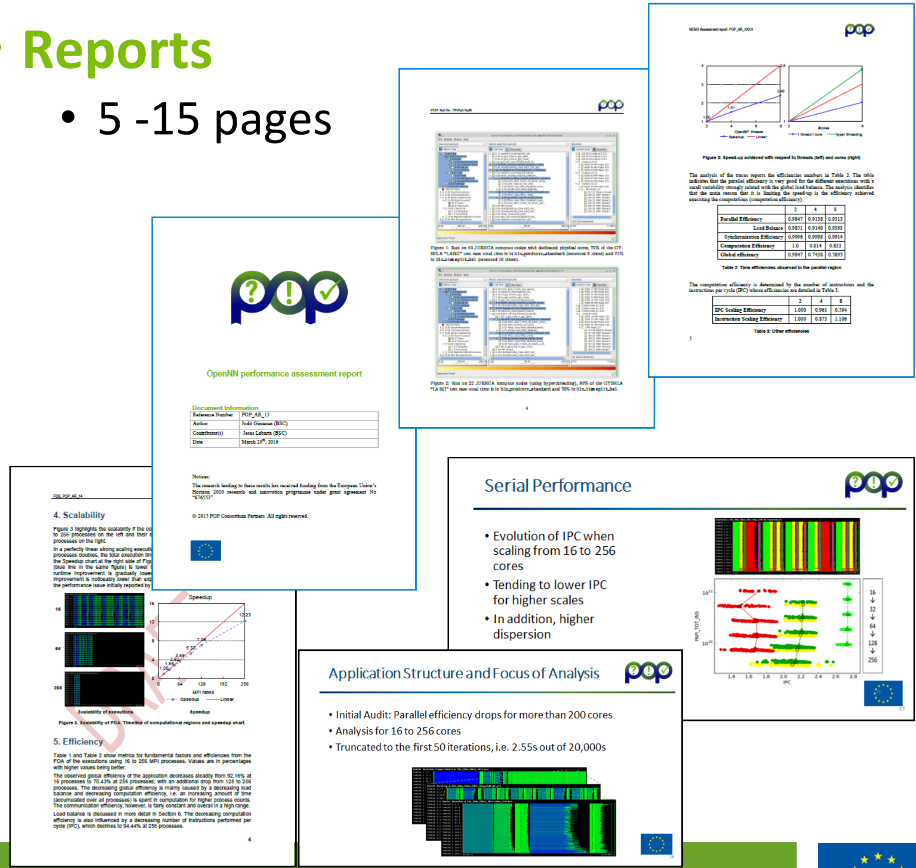
Activities (Dec 2017)

- **195 Services**
  - Completed/reporting: 113
  - Codes being analyzed: 29
  - Waiting user / New: 36
  - Cancelled: 17

- **By type**
  - Audits: 137
  - Plan: 22
  - Proof of concept: 19

+ 5 training workshops

- **Reports**
  - 5 -15 pages



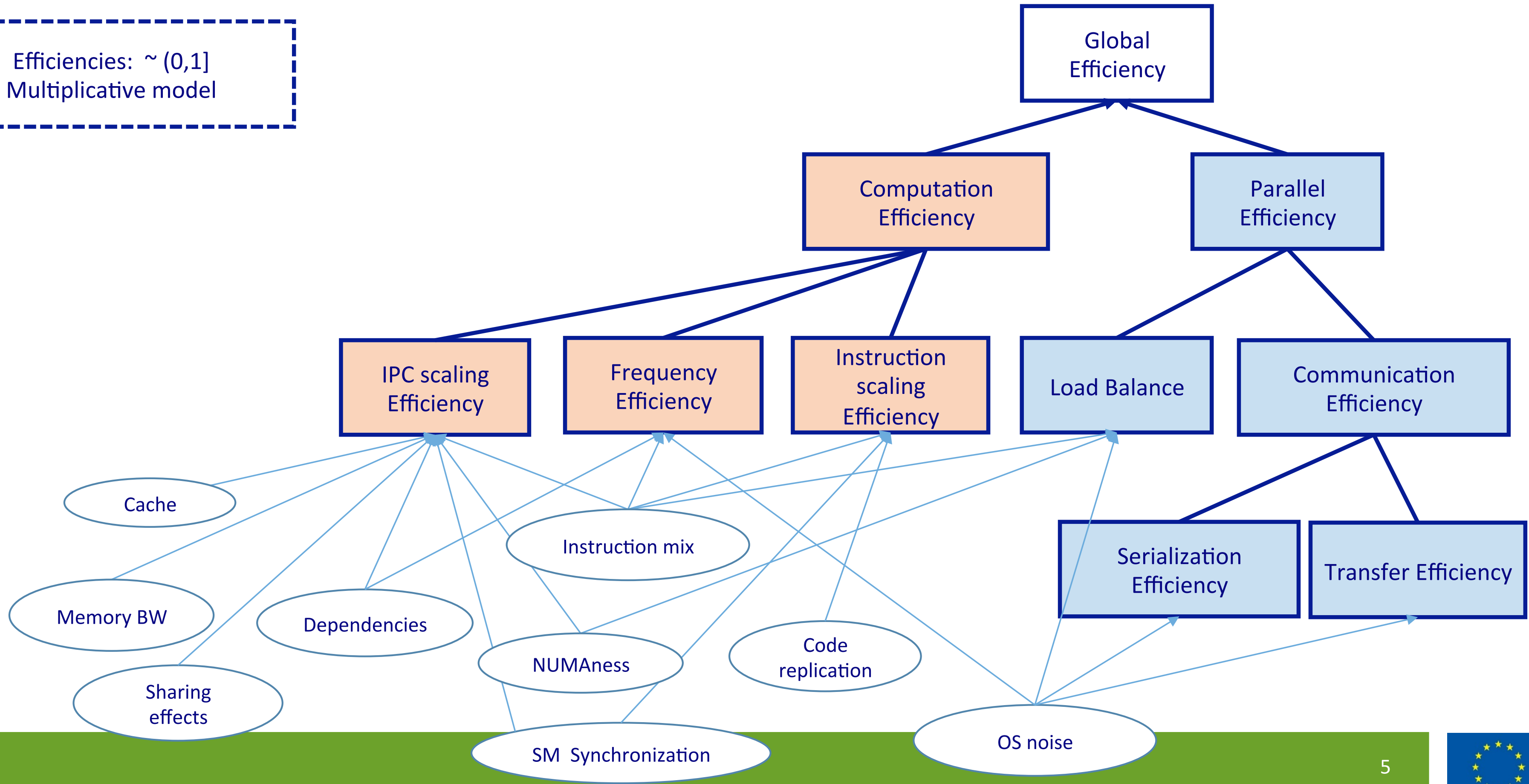
Methodologies and best practices

- Understanding application behaviour
  - Hierarchical performance model
  - Performance Analytics & details
    - Timelines
    - What if
    - Clustering, tracking, folding, …

- Towards productive programming large scale systems
  - MPI - OpenMP interoperability
    - Task based overlap communication and computation
  - Exploiting malleability
    - Dynamic load balance

Hierarchical Performance Model ...

Efficiencies: ~ (0,1]
Multiplicative model



Hierarchical Performance Model …

	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

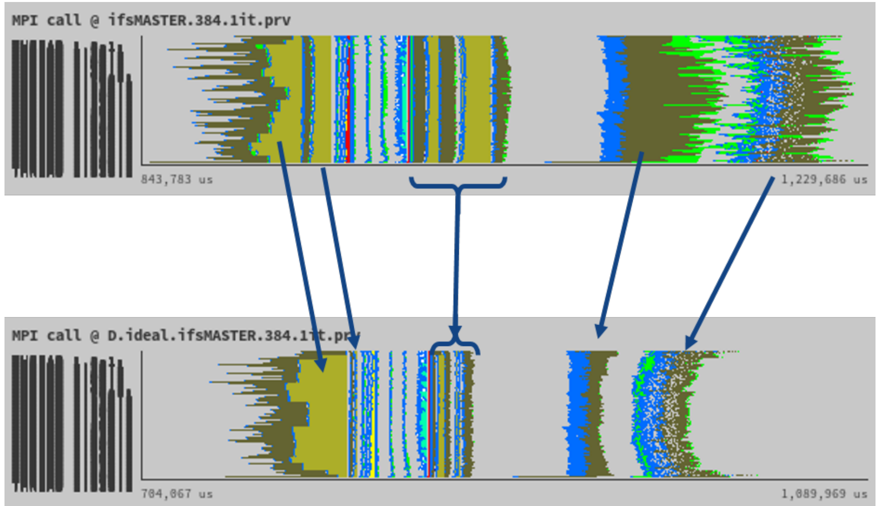
	32	48	64	96	128	256
Parallel Efficiency	0.917	0.906	0.887	0.847	0.864	0.790
Load Balance	0.946	0.925	0.934	0.858	0.871	0.813
Serialization efficiency	0.970	0.980	0.951	0.987	0.994	0.976
Transfer Efficiency	1.000	1.000	1.000	0.999	0.999	0.995
Computation Efficiency	1.000	1.025	1.026	1.036	1.012	0.956
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

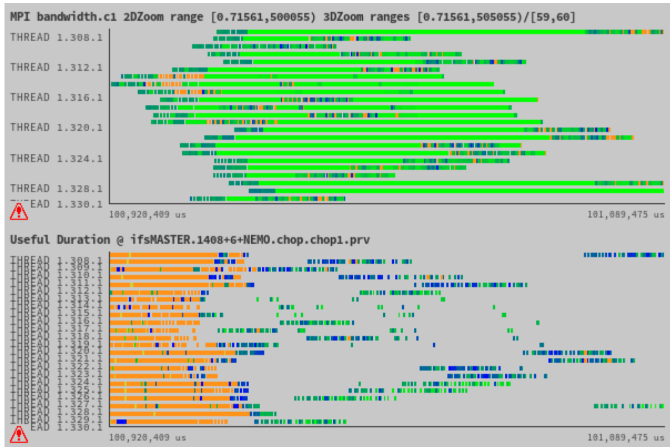
What if MPI had no overhead and transfer was instantaneous ?



Fundamental underlying causes?

How to counteract?

Detailed communication pattern?



Tracking MPI+OMP strong scaling

48x1

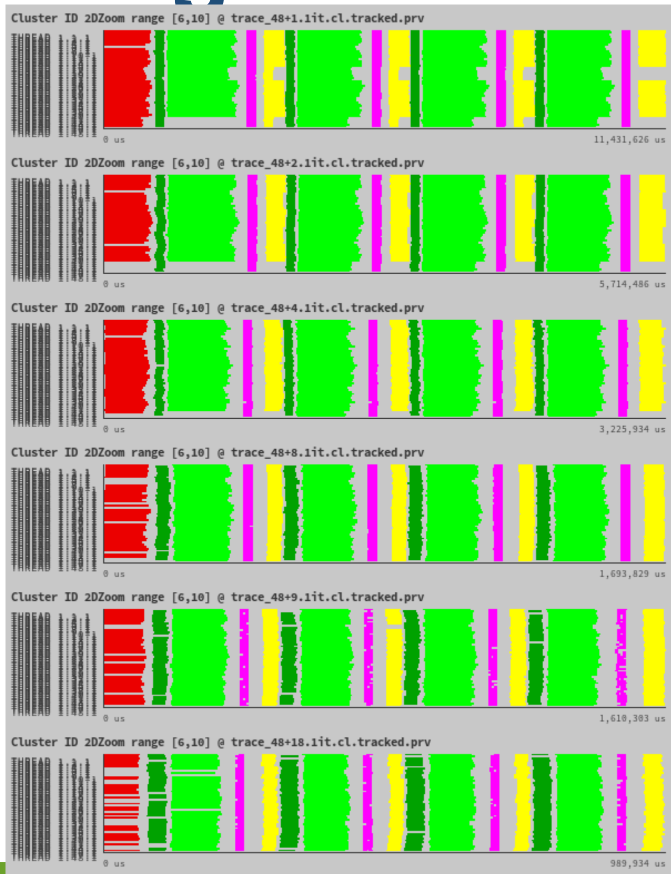
48x2

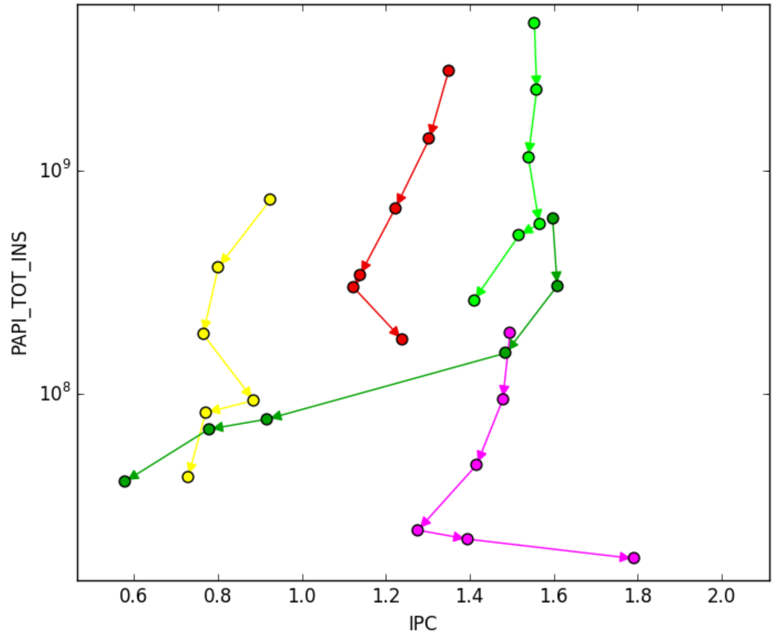
48x4

48x8

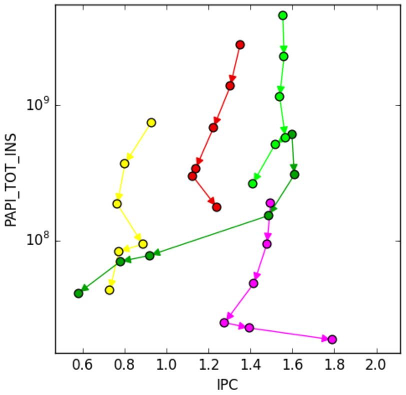
48x9

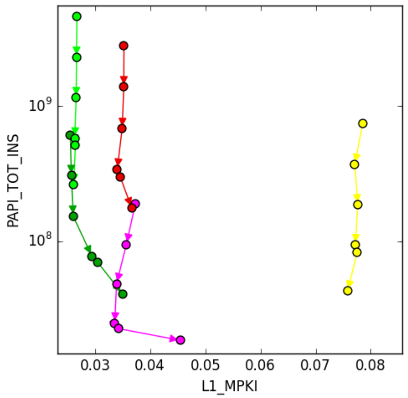
48x18

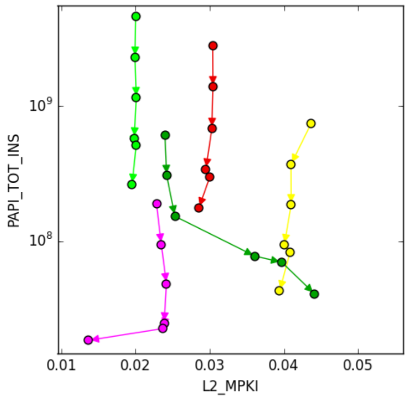


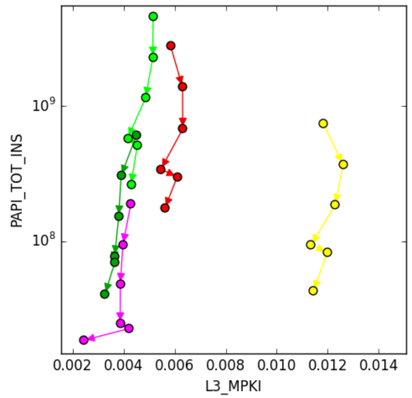


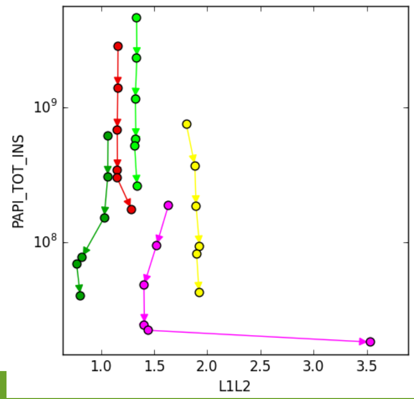
Tracking MPI+OMP strong scaling

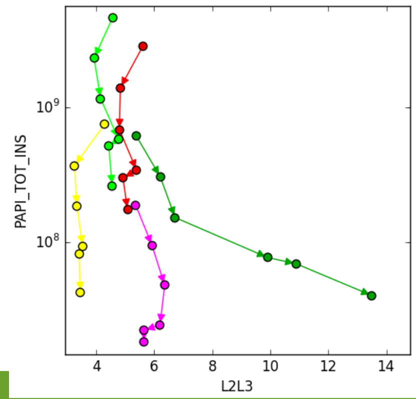






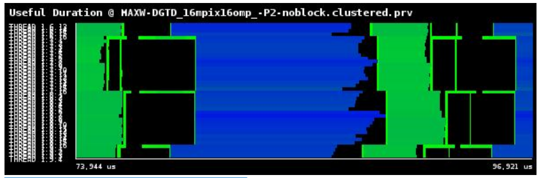






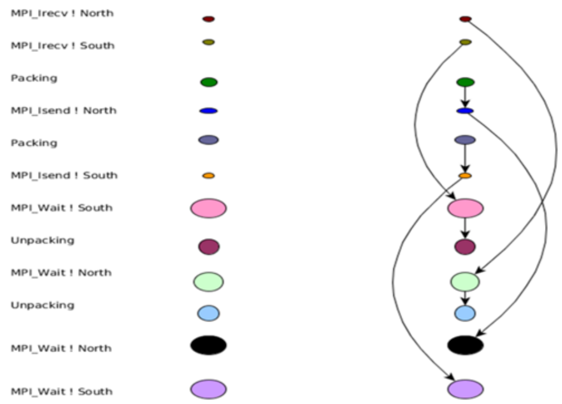
MPI – OpenMP interoperability

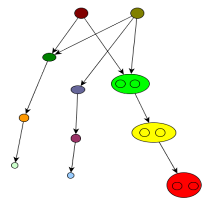
- Hybrid Amdahl's law
  - A fairly "bad message" for programmers
- Significant non parallelized parts
  - pack/unpack
    - Often too fine grain
    - Significant variability
  - MPI calls
    - Too serial
      - Communicator context
      - MPI order semantics
        - Instead of tags
      - Hardwired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



MAXW-DGTD

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

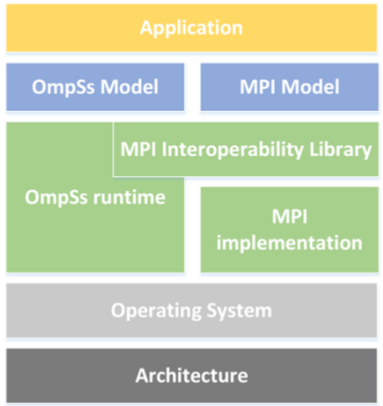


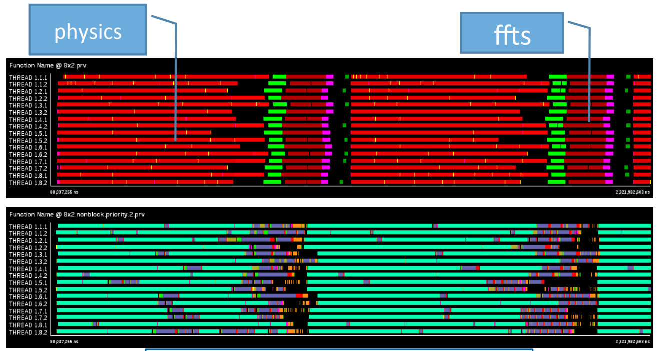


NMMB

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





IFS weather code kernel. ECMWF

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

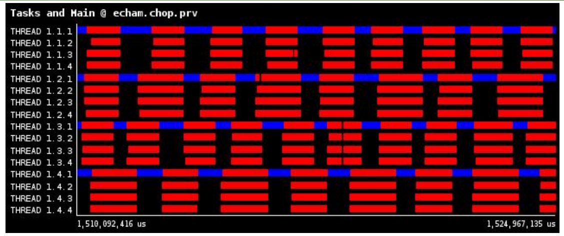
K. Sala et al, "Improving the Interoperability between MPI and Task-Based Programming Models". Submitted

Exploiting malleability

- Dynamic Load Balance & Resource management
  - Intra/inter process/application

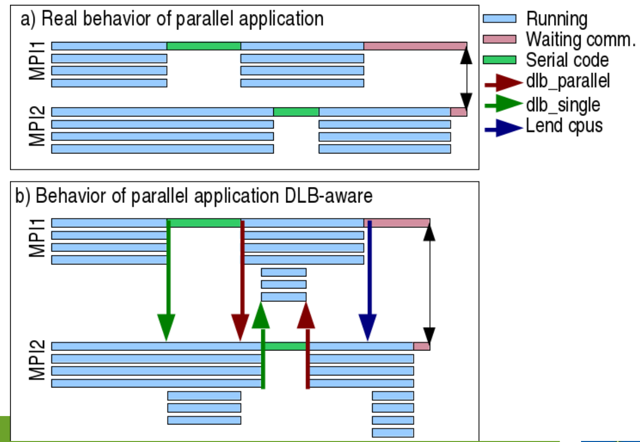
- Library (DLB)
  - Runtime interception (MPIP, OMPT, …)
  - API to hint resource demands
  - Core reallocation policy

- Opportunity to fight Amdalh's law
  - Productive / Easy !!!
    - Nx1
    - Hybridize imbalanced regions



ECHAM

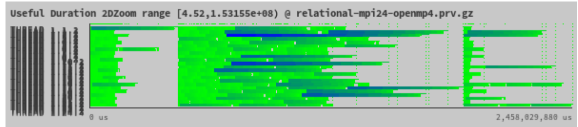
dynamic load balancing

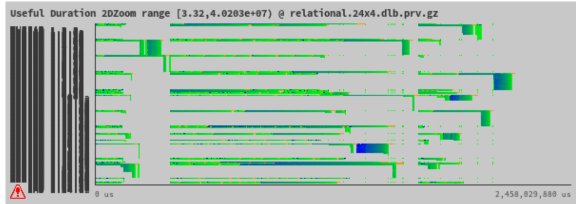


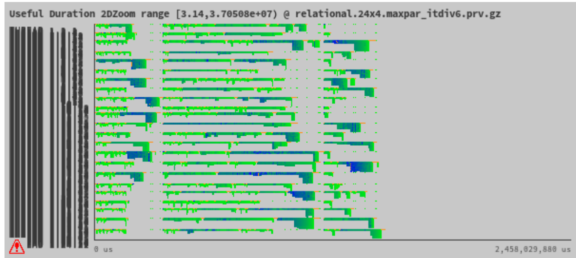
"LeWI: A Runtime Balancing Algorithm for Nested Parallelism". M.Garcia et al. ICPP09
"Hints to improve automatic load balancing with LeWI for hybrid applications" JPDC2014

Exploiting malleability

- 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 !!!
    - Nx1
    - Hybridize imbalanced regions





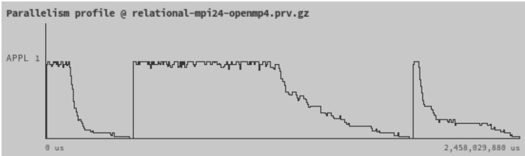


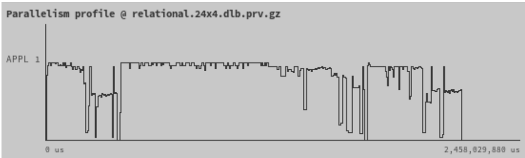
Relational Discovery

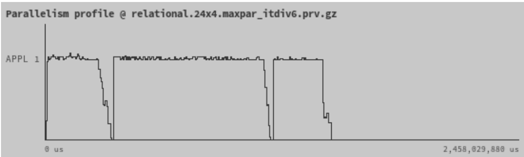
"LeWI: A Runtime Balancing Algorithm for Nested Parallelism". M.Garcia et al. ICPP09
"Hints to improve automatic load balancing with LeWI for hybrid applications" JPDC2014

Exploiting malleability

- 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 !!!
    - Nx1
    - Hybridize imbalanced regions





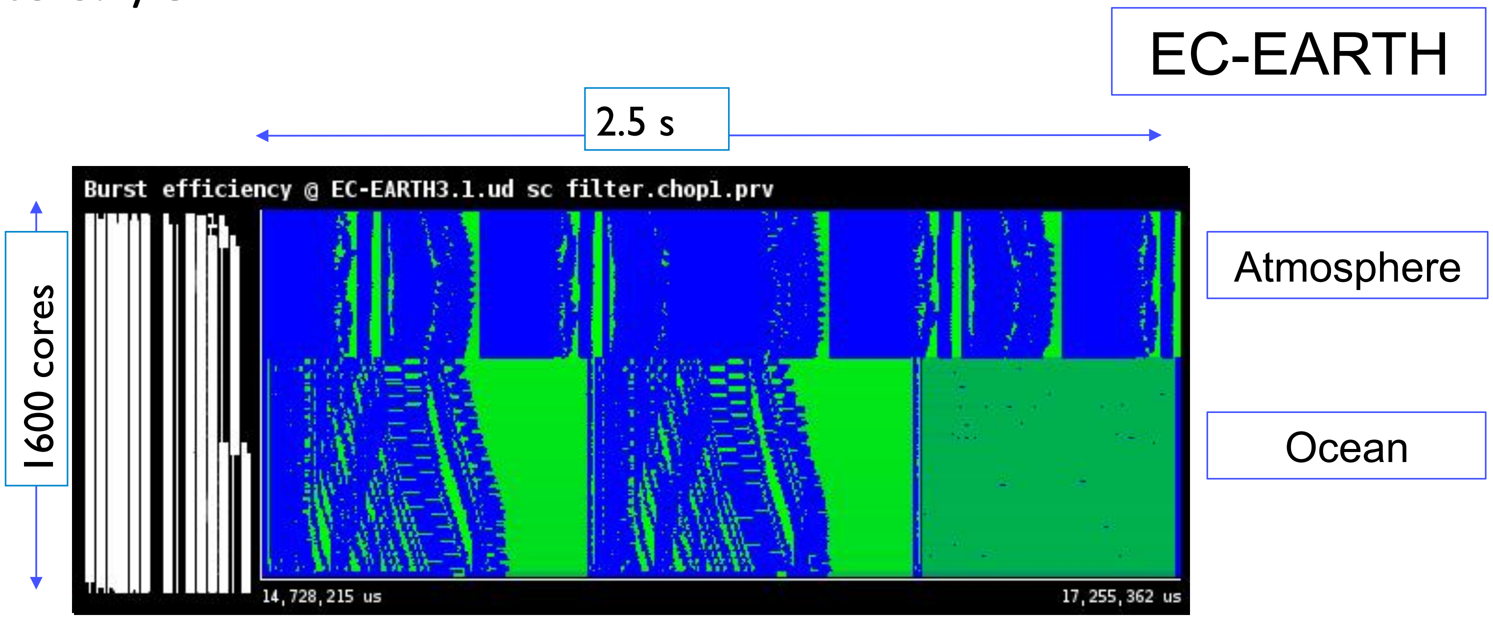


Relational Discovery

"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

Coupled codes

- Multiple physics, domains
- Compute & I/O

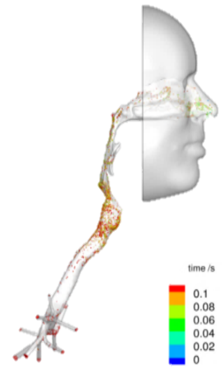


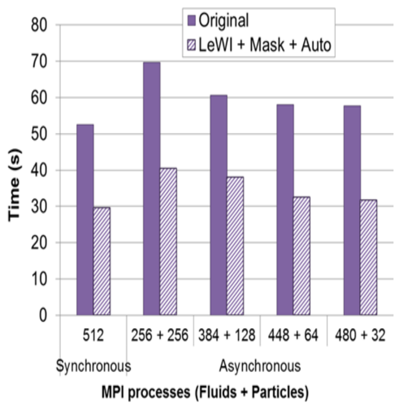
26.7MB trace

Eff: 0.43; LB: 0.52; Comm:0.81

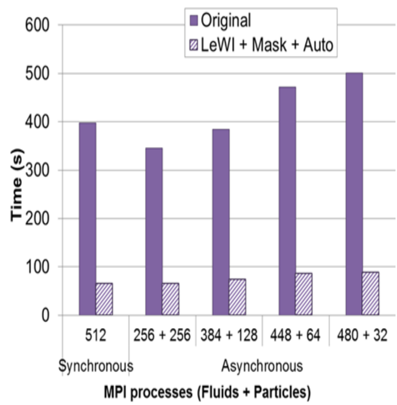
Exploiting malleability @ Coupled codes

- Dynamic load balance
  - How to allocate resources ? Configure the runs
  - Important to “maximize” performance …
  - … without needing to care about detailed configuration

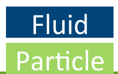




Fluid dominated



Particle dominated



Closing remarks

- The real parallel programming revolution
  - … is in the mindset of programmers
    - From latency to throughput oriented !!!
    - Think global, specify local
  - … and can be achieved productively
    - Incrementally
    - On a standard programming model (MPI+OpenMP)

- 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

POP

- Past
  - Huge effort, high appreciation
  - Provided useful insight to a large set of users
  - Using “simple” techniques
- Plan
  - Continue with basic service
  - Ease of use of tools
  - Extend use of more advanced techniques (clustering, tracking, folding,…)
  - Emphasis on programming best practices
  - Towards larger scales

Performance Optimisation and Productivity

A Centre of Excellence in Computing Applications

Contact:

https://www.pop-coe.eu

mailto:pop@bsc.es

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 676553.