

allinea



Leaders in parallel software development tools


Performance, energy...

How to improve the efficiency of
HPC workloads ?



Agenda



- HPC and Efficiency: different perspectives
 - How to reach the highest levels of efficiency with Allinea?
 - Live demonstration
 - Summary
- 

Allinea

The Company

- **HPC tools company since 2002**
 - Leading in HPC software tools market worldwide
 - Global customer base
- **Helping the HPC community design the best applications**
 - Unrivalled productive and easy-to-use development environment...
 - ... To help reach the highest level of performance and scalability
- **Helping HPC users make the most of their allocations**
 - Unique solutions to understand application performance
 - Innovative approach to resolve cutting-edge challenges

What is efficiency?

Example : Formula 1 rules

Season 2014

(http://www.formula1.com/inside_f1/rules_and_regulations/sporting_regulations/8713/fia.html)

3. Combined Restricted Wind Tunnel Testing and Restricted CFD Simulation Restriction

3.1 The usage limits for Restricted Wind Tunnel Testing and Restricted CFD Simulations are expressed in terms of Wind On Time, number of runs, tunnel occupancy and **CFD Teraflops Usage during an Aerodynamic Testing Period.**

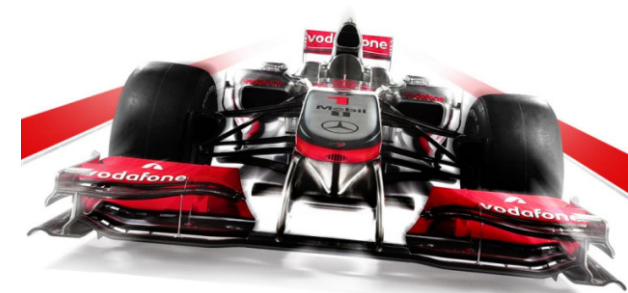
[...]

3.4 CFD Teraflops Usage is defined as the average number of teraflops of computing power used for the purpose of making Restricted CFD Simulations during the Aerodynamic Testing Period.

How to optimize a code from this perspective?

How to run a code from this perspective?

How about the future rules?



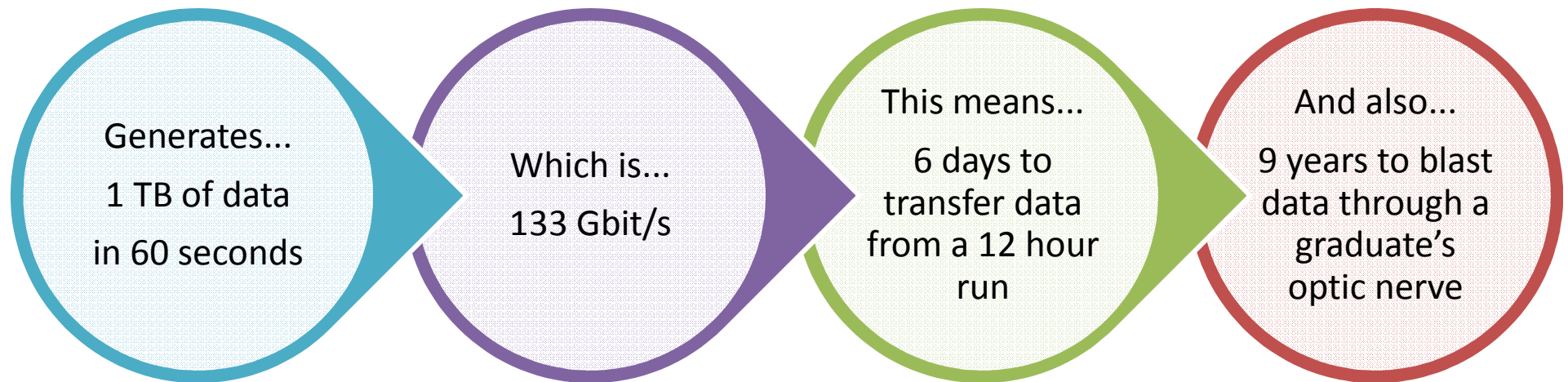
Allinea Unified environment

- A modern integrated environment for HPC developers
- Supporting the lifecycle of application development and improvement
 - Allinea DDT : Productively debug code
 - Allinea MAP : Enhance application performance
- Designed for productivity
 - Consistent easy to use tools
 - Enables effective HPC development
- Improve system usage
 - Fewer failed jobs
 - Higher application performance



Beware exploding bandwidth needs...

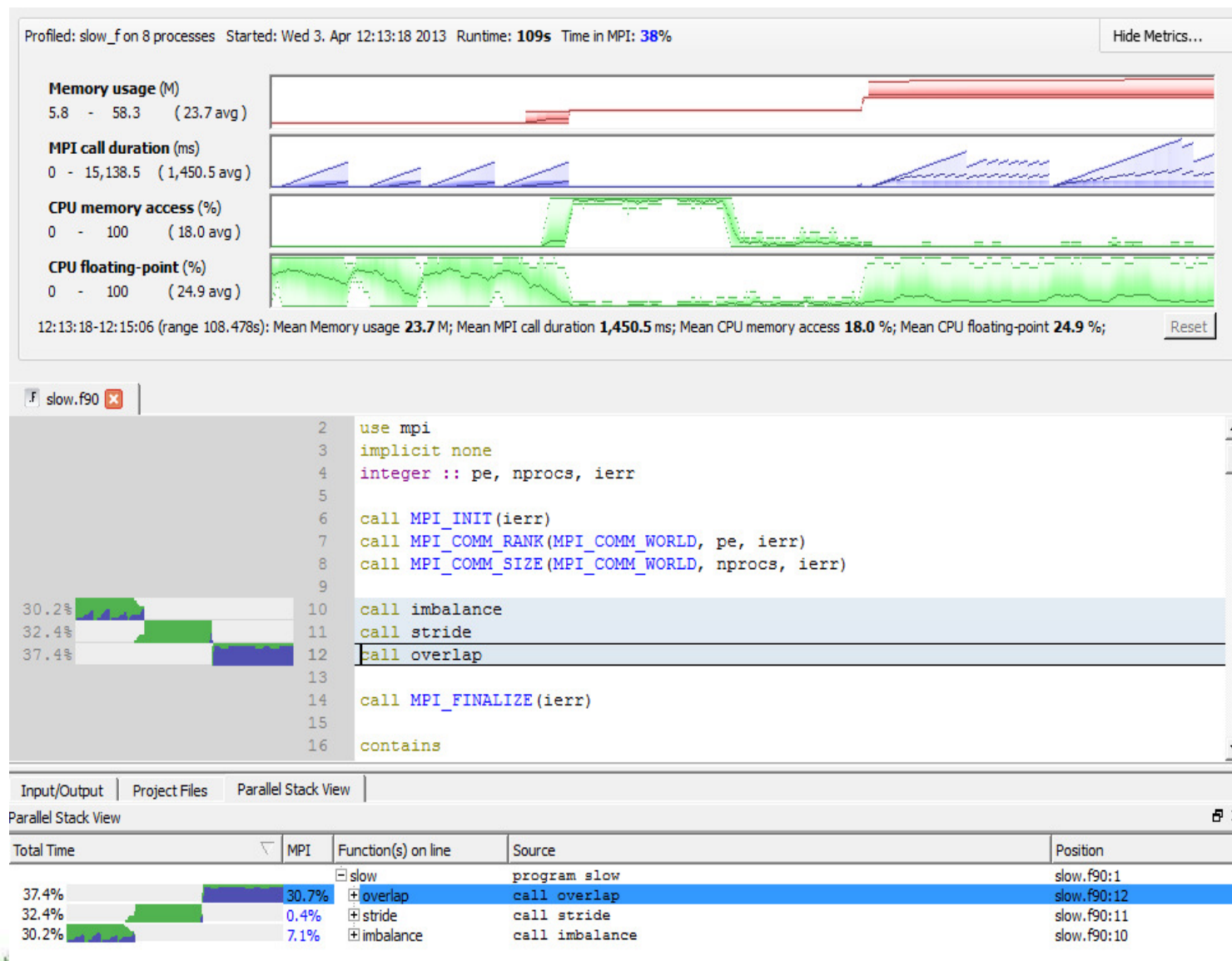
Trivial 16k processes wave equation code



```
void do_math(int i)
{
    const double dtime = 0.3;
    const double c = 1.0;
    const double dx = 1.0;
    double tau, sqtau;

    tau = (c * dtime / dx);
    sqtau = tau * tau;
    newval[i] = (2.0 * values[i]) - oldval[i]
        + (sqtau * (values[i-1] - (2.0 * values[i]) + values[i+1]));
}
```


Attacking Visual Scalability



Common horizontal axis



Aggregate across all processes and threads



Highlight imbalance visually



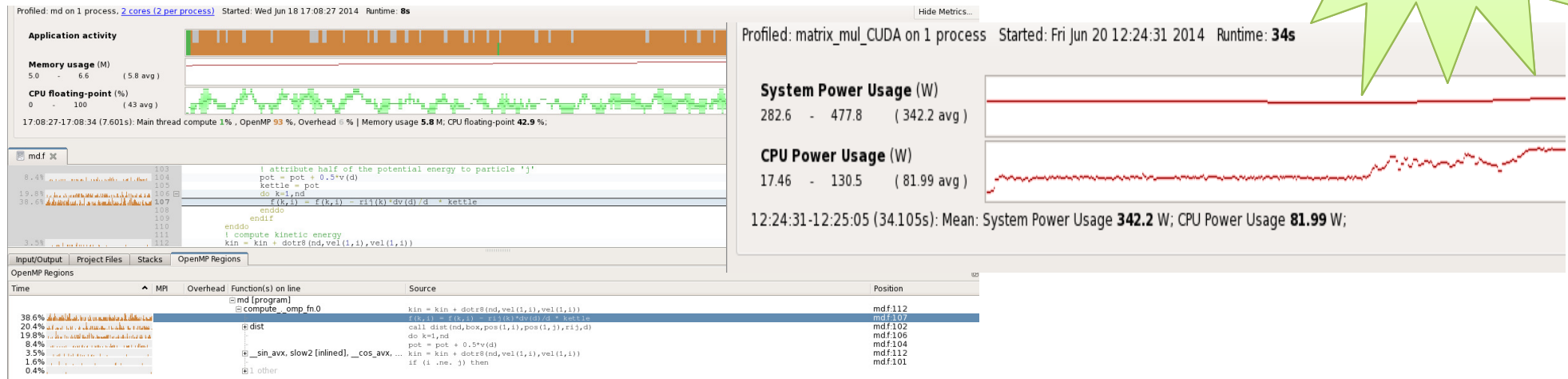
Always refer to source code



Optional: Target specific areas for instrumentation

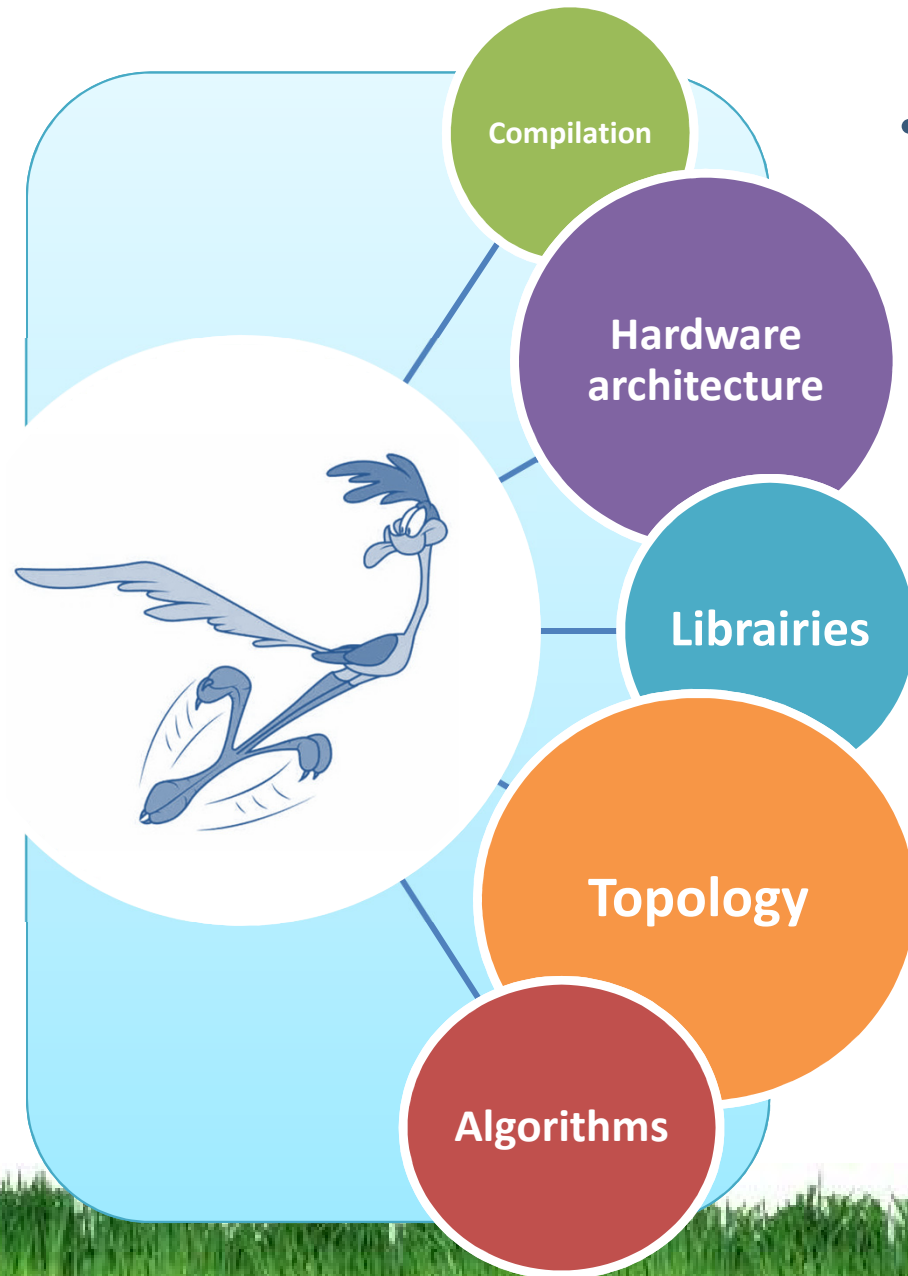
New in Allinea MAP 4.3

NEW



- **Full support for multithreaded & hybrid codes**
- **Now integrates energy metrics !**
 - Find the energy consuming hotspots in your application to tune them
 - Reduce the running costs of your application by minimizing energy footprint
- **Now adding accelerator metrics !**
 - Evaluate the impact of your accelerators in your code
 - Reach unrivalled Flop/watt with your accelerated applications

HPC is not only about development



- **HPC combines several specialities**

- Science (physics, biology...)
- Computer science
- Numerical analysis

How to combine those independent areas of expertise to increase the “*efficiency*” of applications ?

Understand cluster usage efficiency

- **Monitors application behavior to provide answers:**
 - Are the applications running on the cluster “efficient” ?
 - Are there software or hardware bottlenecks affecting performance ?
 - Is the combination of application parameters/libraries optimal ?
 - What cluster/scale should the user choose for his job ?

- **Effortless one-touch reports:**

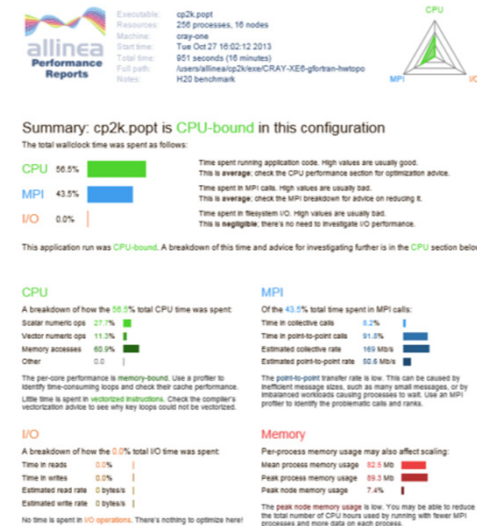
```
mpirun -n 42 ./my_executable argument1
```

becomes

```
perf-report mpirun -n 42 ./my_executable argument1
```

- **Available on HP systems – with unique added value !**

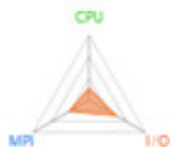
- Implementation of a connector with CMU in development



Designed for better runs, quickly



Executable: madbench
Resources: 9 processes, 1 node
Machine: Namaste-II
Start time: Mon Nov 4 12:26:45 2013
Total time: 11 seconds (0 minutes)
Full path: /tmp/madbench
Notes:



Summary: madbench is I/O-bound in this configuration

The total wallclock time was spent as follows:

CPU 17.9%

MPI 34.5%

I/O 47.6%

Time spent running application code. High values are usually good. This is **very low**; focus on improving MPI or I/O performance first.

Time spent in MPI calls. High values are usually bad.

This is **average**; check the MPI breakdown for advice on reducing it.

Time spent in filesystem I/O. High values are usually bad.

This is **high**; check the I/O breakdown section for optimization advice.

This application run was **I/O-bound**. A breakdown of this time and advice for investigating further is in the I/O section below.

CPU

A breakdown of how the 17.9% total CPU time was spent:

Scalar numeric ops 15.0%

Vector numeric ops 0.0%

Memory accesses 85.0%

Other 0.0%

The per-core performance is **memory-bound**. Use a profiler to identify time-consuming loops and check their cache performance.

MPI

Of the 34.5% total time spent in MPI calls:

Time in collective calls 100.0%

Time in point-to-point calls 0.0%

Estimated collective rate 40.8 bytes/s

Estimated point-to-point rate 0.00 bytes/s

Most of the time is spent in **collective calls** with a very low transfer rate. This suggests load imbalance is causing synchronization overhead; use an MPI profiler to investigate further.

I/O

A breakdown of how the 47.6% total I/O time was spent:

Time in reads 24.5%

Time in writes 75.5%

Estimated read rate 400 Mb/s

Estimated write rate 70.0 Mb/s

Most of the time is spent in **write operations** with a low effective transfer rate. This may be caused by contention for the filesystem or inefficient access patterns. Use an I/O profiler to investigate which write calls are affected.

Memory

Per-process memory usage may also affect scaling:

Mean process memory usage 255 Mb

Peak process memory usage 808 Mb

Peak node memory usage 15.2%

The **peak node memory usage** is very low. You may be able to reduce the total number of CPU hours used by running with fewer MPI processes and more data on each process.

No instrumentation needed

No source code needed

No recompilation needed

Less than 5% runtime overhead

Fully scalable

Run regularly – or in regression tests

Explicit and usable output

New in version 4.3



NEW

Energy

CPU energy report:

Total energy	1572.08 J	<div></div>
Peak power	76.00 W	<div></div>
Average power	63.47 W	<div></div>


Whole system energy report:

Total energy	7707.29 J	<div></div>
Peak power	334.00 W	<div></div>
Average power	311.17 W	<div></div>

- **Now integrates energy metrics !**
 - Understand the energy consumption of your application
 - Reduce the running costs of your application by minimizing energy footprint
- **Full support for multithreaded & hybrid codes**

Summary



- **Efficiency has different meanings depending on the context**
 - Number of Flops, runtime, energy consumption, Flops/watt...
 - “Green” hardware is best without “green” applications
 - Tools need to provide the general view and answer both existing and cutting-edge needs
 - **Develop codes for your efficiency with Allinea MAP**
 - Provides details about your application: CPU, MPI, memory, disk I/Os...
 - ... and now Accelerators and Energy metrics to reduce operating costs even further !
 - Supports a wide range of applications (MPI, OpenMP, accelerated, hybrid...)
 - **Make the most of clusters with Allinea Performance Reports**
 - One-touch reports to understand the application behaviour and performance
 - Reduce the clusters operating costs by using energy metrics at the application level
- 

allinea



Leaders in parallel software development tools

Thank you !

Your contacts :

- Technical Support team :
- Sales team :

support@allinea.com

sales@allinea.com

