



High performance tools to debug, profile, and analyze your applications

What's new in Allinea's tools

From easy batch script integration & remote access
to energy profiling



Introduction Agenda

- **Overview of HPC current and future needs**
- **What's new in Allinea's tools**
 - Transitioning from CPU Hours to Power Usage
 - Enabling easy remote application development
- **What is coming next**
 - Reducing trapped capacity on next-gen processors
 - Coming Allinea "good stuff"

HPC Ultimate target



Example: Weather and Forecasting models



Demands are evolving



Scalability

- Enable multi-physics simulations
- Run larger, more accurate models
- Resolve ground-breaking scientific problems

Efficiency

- Maximize science output per \$
- Minimize time to result
- Monitor and reduce wasted resources (energy..)

Simplicity

- Readiness of applications on HPC platforms
- Minimize learning curve for HPC users
- Facilitate dialogue with scientific communities

Allinea's vision

- **Helping maximize HPC production**

- Reduce HPC systems operating costs
- Resolve cutting-edge challenges
- Promote Efficiency (as opposed to Utilization)
- Transfer knowledge to HPC communities

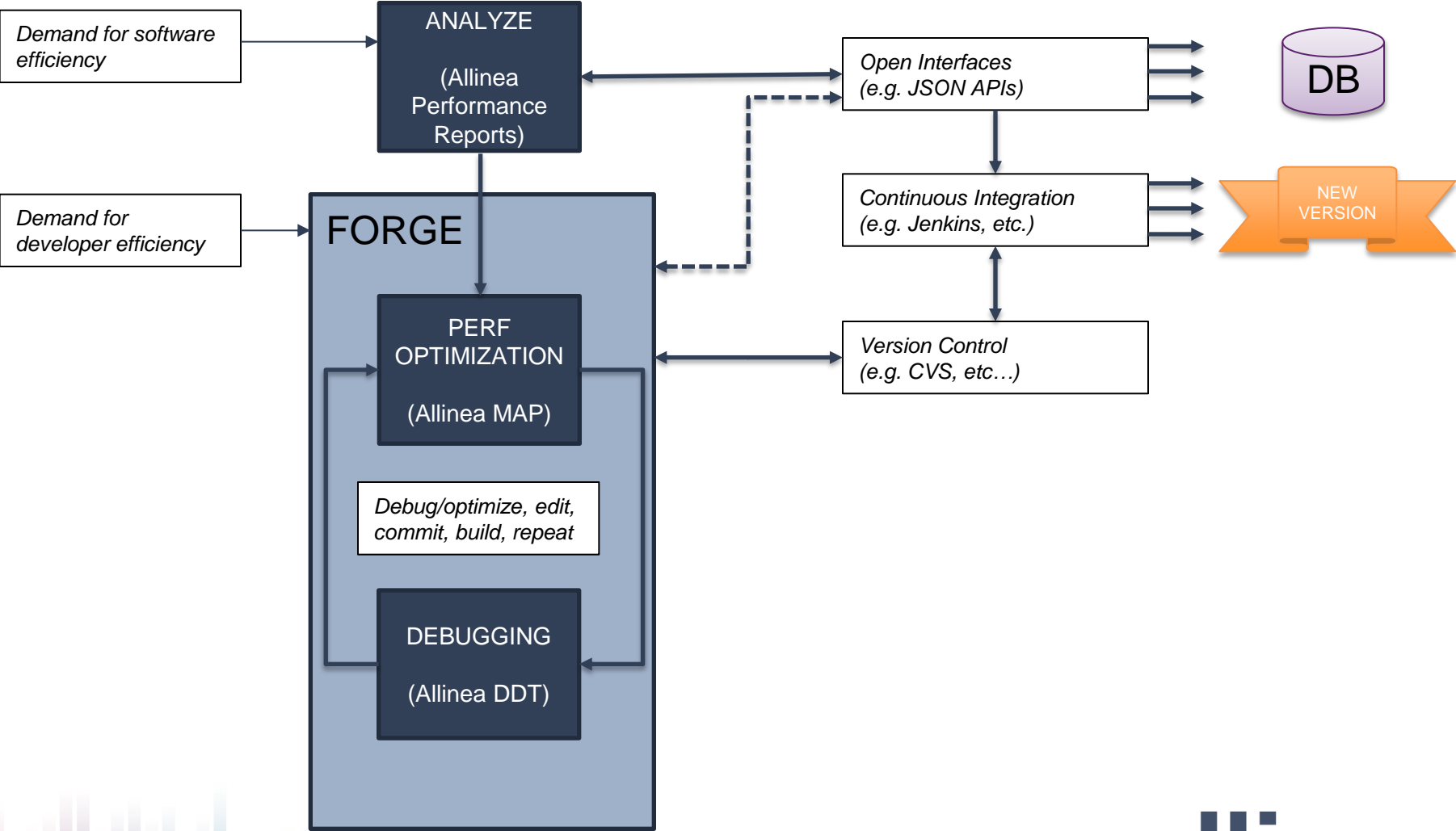


- **Helping the HPC community design the best applications**



- Reach highest levels of performance and scalability
- Improve scientific code quality and accuracy

Development process: tools-centric view



Allinea MAP and CrayPAT : a great synergy

Simple optimization with Allinea MAP

- Characterize performance at-scale with a simple GUI tool
- See which lines of code are hotspots
- Identify common problems at once

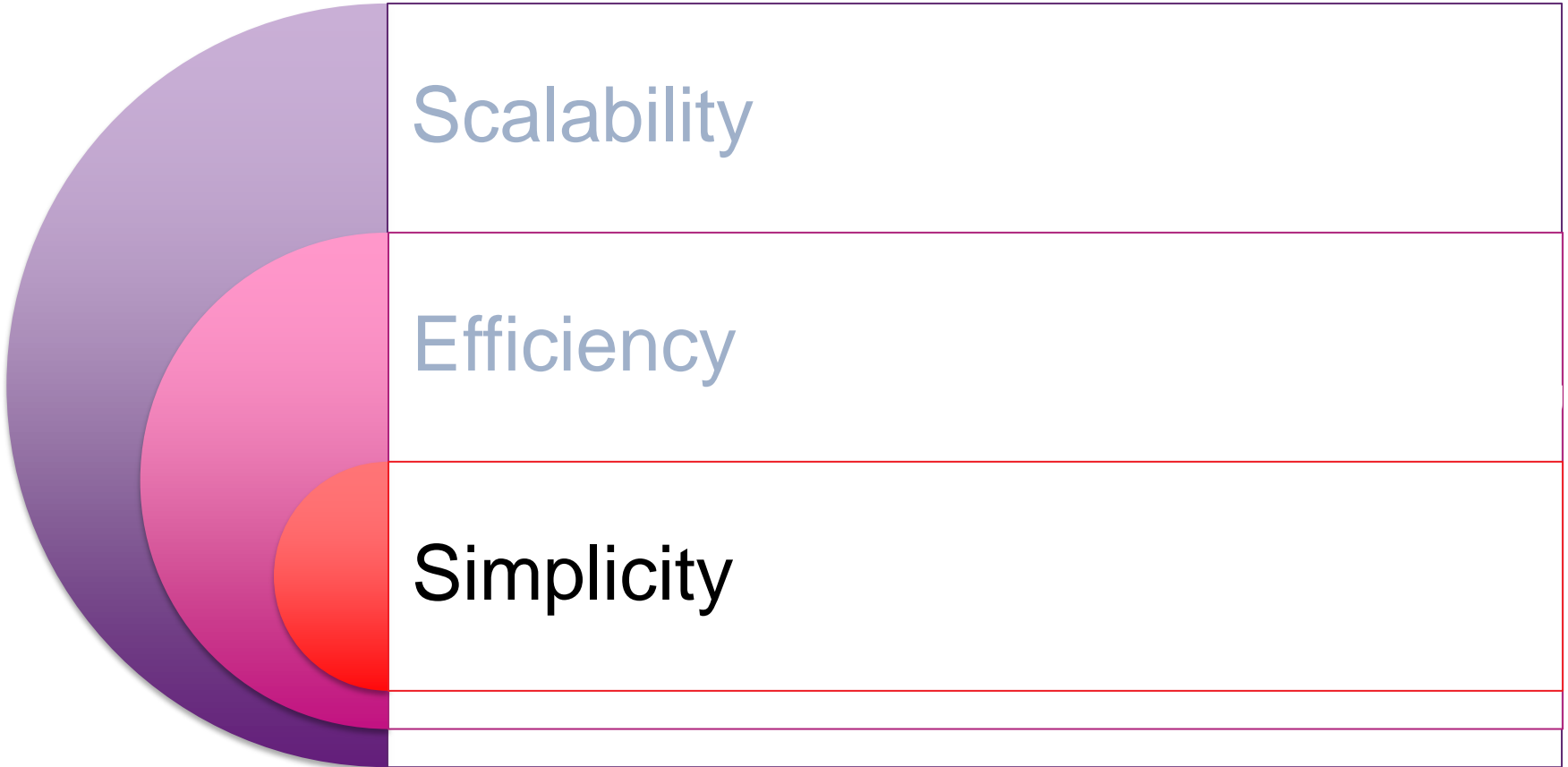
Prepare optimization strategy with Allinea MAP

- Document performance issues to communicate to CrayPAT experts
- Identify loop(s) to instrument with CrayPAT API
- Identify CrayPAT performance counter(s) to record

Fine tune the code with CrayPAT

- Retrieve low-level details with CrayPAT
- Fix up CPU usage to make the code fly

Demands are evolving



Allinea Remote Client: fast interactive debugging



RUN

Run and debug a program.

ATTACH

Attach to an already running

OPEN CORE

Open a core file from a previous

MANUAL LAUNCH (ADVANCED)

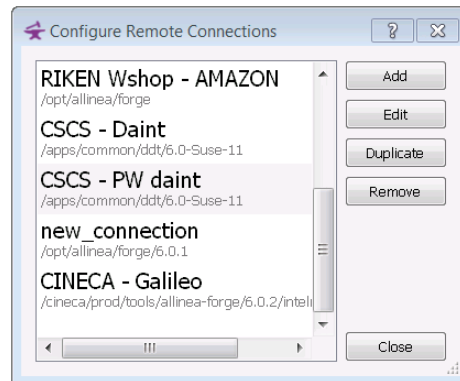
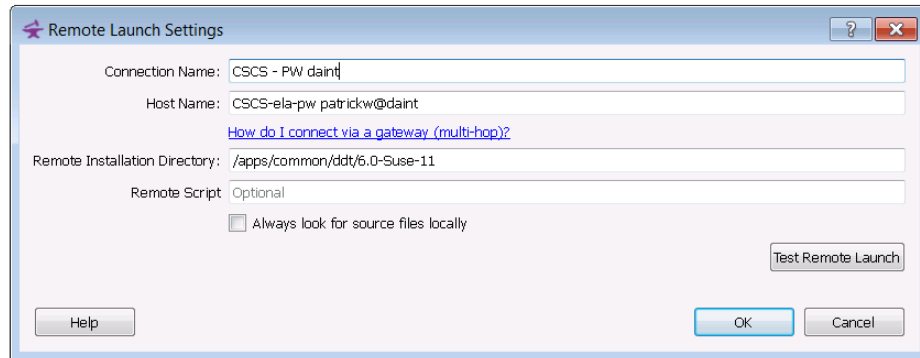
Manually launch the backend

OPTIONS

Remote Launch:

Configure ...

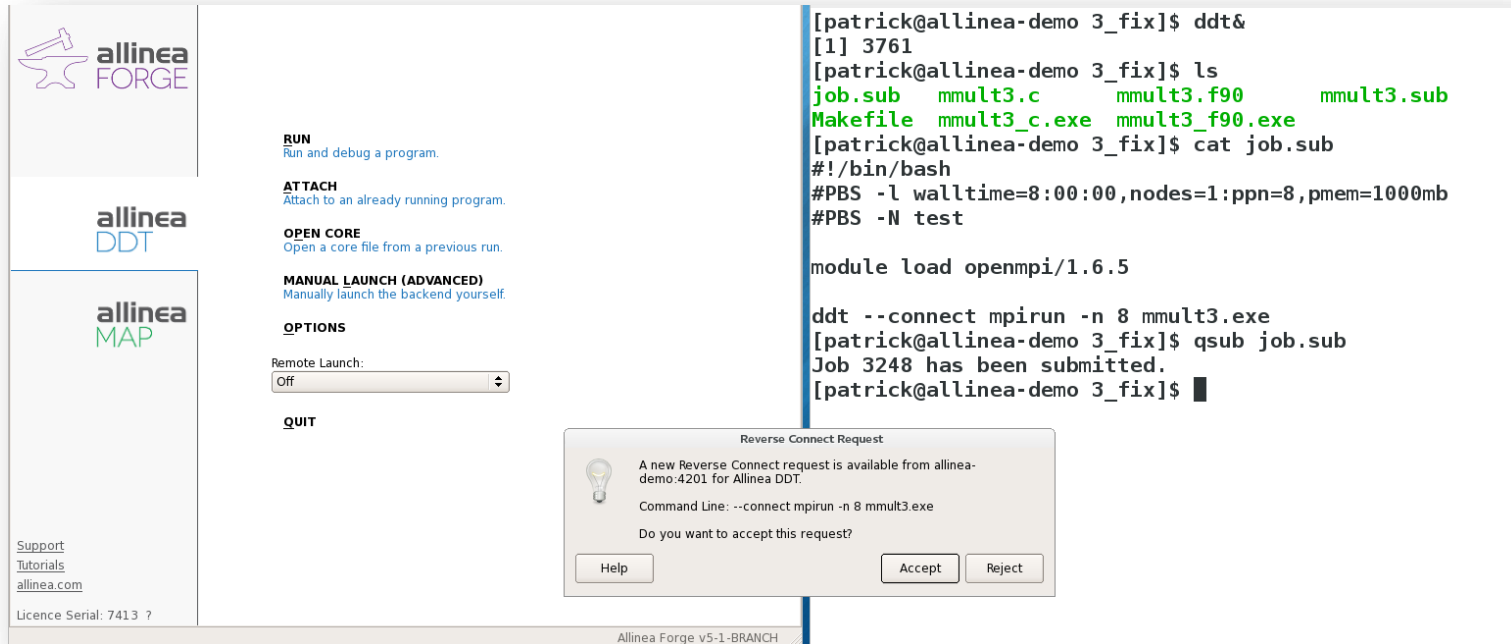
QUIT



Support
Tutorials
allinea.com

Remote Client ?

Reverse connect: the end of template files



The screenshot displays the Alinea Forge v5-1-BRANCH interface. On the left, there is a sidebar with the Alinea Forge logo and navigation options for allinea DDT and allinea MAP. The main area is divided into two panes. The left pane contains a menu with options: RUN (Run and debug a program.), ATTACH (Attach to an already running program.), OPEN CORE (Open a core file from a previous run.), MANUAL LAUNCH (ADVANCED) (Manually launch the backend yourself.), OPTIONS (Remote Launch: Off), and QUIT. The right pane shows a terminal window with the following commands and output:

```
[patrick@allinea-demo 3_fix]$ ddt&
[1] 3761
[patrick@allinea-demo 3_fix]$ ls
job.sub  mmult3.c  mmult3.f90  mmult3.sub
Makefile mmult3_c.exe mmult3_f90.exe
[patrick@allinea-demo 3_fix]$ cat job.sub
#!/bin/bash
#PBS -l walltime=8:00:00,nodes=1:ppn=8,pmem=1000mb
#PBS -N test

module load openmpi/1.6.5

ddt --connect mpirun -n 8 mmult3.exe
[patrick@allinea-demo 3_fix]$ qsub job.sub
Job 3248 has been submitted.
[patrick@allinea-demo 3_fix]$ █
```

A dialog box titled "Reverse Connect Request" is overlaid on the terminal. It contains a lightbulb icon and the following text:

A new Reverse Connect request is available from allinea-demo:4201 for Allinea DDT.
Command Line: --connect mpirun -n 8 mmult3.exe
Do you want to accept this request?

The dialog box has three buttons: "Help", "Accept", and "Reject".

Support
Tutorials
allinea.com
Licence Serial: 7413 ?
Allinea Forge v5-1-BRANCH

Towards better efficiency with Alinea's tools

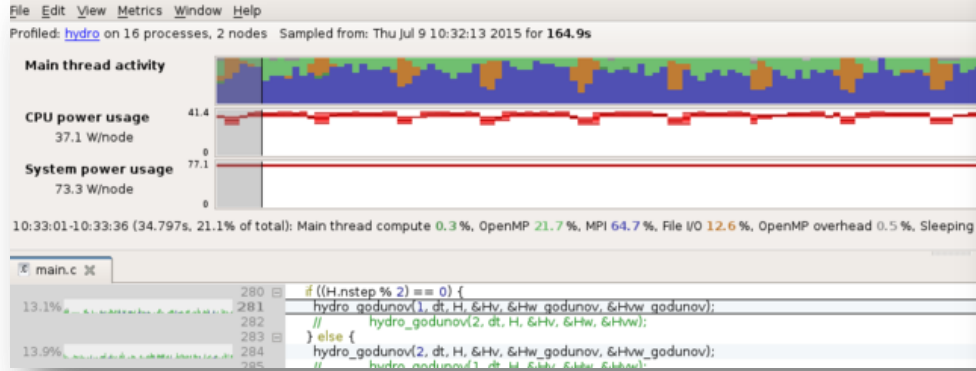


Scalability

Efficiency

Simplicity

Energy efficiency with Alinea's tools



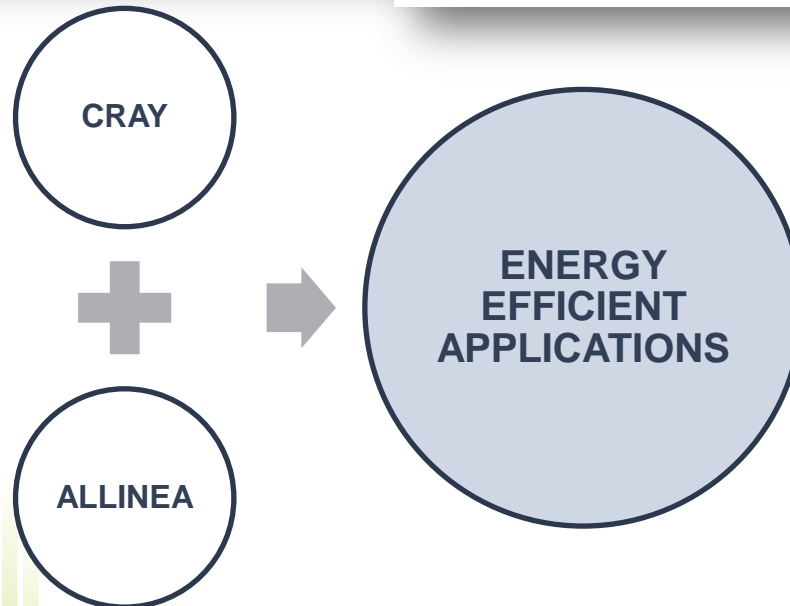
Energy

A breakdown of how the 3.6 Wh was used:

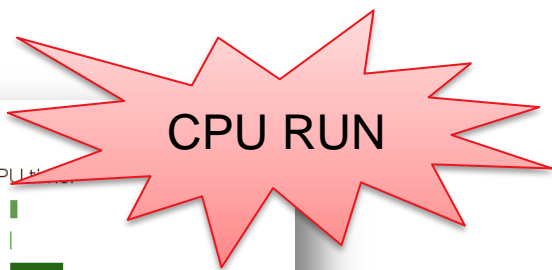
CPU	62.9%	
System	37.1%	
Mean node power	92.4 W	
Peak node power	94 W	

Significant energy is wasted during MPI communications. It may be more efficient to use fewer nodes with more data on each node.

Significant time is spent waiting for memory accesses. Reducing the CPU clock frequency could reduce overall energy usage.



Quantify gains immediately



CPU

A breakdown of the 94.6% CPU time:

Scalar numeric ops	11.7%	
Vector numeric ops	0.0%	
Memory accesses	88.2%	█
Waiting for accelerators	0.0%	

The per-core performance is **memory-bound**. Use a profiler to identify time-consuming operations.

No time is spent in vectorization advice.

Energy

A breakdown of how the 3.6 Wh was used:

CPU	62.9%	█
System	37.1%	█
Mean node power	92.4 W	█
Peak node power	94 W	█

Significant energy is wasted during MPI communications. It may be more efficient to use fewer nodes with more data on each node.

Significant time is spent waiting for memory accesses. Reducing the CPU clock frequency could reduce overall energy usage.

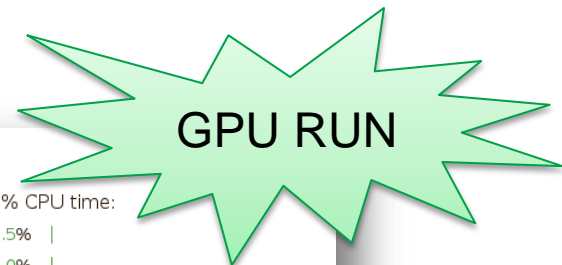
Accelerators

A breakdown of how accelerators were used:

GPU utilization	0.0%	
Global memory accesses	0.0%	
Mean GPU memory usage	0.0%	
Peak GPU memory usage	0.0%	

GPUs are available but are not used. Identify suitable hot loops with a profiler and try offloading them to the accelerator.

The peak device memory usage is low. It may be more efficient to offload a larger portion of the dataset to each device.



CPU

A breakdown of the 70.2% CPU time:

Scalar numeric ops	2.5%	
Vector numeric ops	0.0%	
Memory accesses	39.7%	█
Waiting for accelerators	61.7%	█

Most of the time is spent waiting for accelerators. Use asynchronous calls to avoid CPU stalls.

The per-core performance is **memory-bound**. Use a profiler to identify time-consuming operations.

Energy

A breakdown of how the 2.84 Wh was used:

CPU	28.4%	█
System	71.6%	█
Mean node power	175.8 W	█
Peak node power	163 W	█

Energy usage appears to be optimal.

Accelerators

A breakdown of how accelerators were used:

GPU utilization	92.5%	█
Global memory accesses	40.4%	█
Mean GPU memory usage	9.6%	
Peak GPU memory usage	15.2%	

Significant time is spent in global memory accesses. Try modifying kernels to use shared memory instead and check for bad striding patterns.

The peak device memory usage is low. It may be more efficient to offload a larger portion of the dataset to each device.

Custom metrics with Allinea MAP

File Edit View Metrics Window Help

Profiled: [mmult1_c.exe](#) on 8 processes, 1 node Sampled from: Thu Feb 11 12:30:51 2016 for **95.8s** Hide Metrics..

Main thread activity

Interrupts
2.81 k/s

12:30:51-12:32:26 (95.759s): Main thread compute **85.6 %**, MPI **14.4 %**, File I/O **0.0 %**, Sleeping **0 %** | Interrupts **2.81 k/s** Zoom

mmult1.c x

```
55 for(int i=0; i<size/nslices; i++)
56 {
57     for(int j=0; j<size; j++)
58     {
59         double res = 0.0;
60
61         for(int k=0; k<size; k++)
62         {
63             res += A[i*size+k]*B[k*size+j];
64         }
65         C[i*size+j] += res;
66     }
67 }
68
69
70
```

Time spent on line 63

Breakdown of the 84.1% time spent on this line:

- Executing instructions 100.0%
- Calling other functions 0.0%

Time in instructions executed:

- Scalar floating-point 0.0%
- Vector floating point 3.8%
- Scalar integer 1.6%
- Vector integer 0.0%
- Memory access* 95.5%
- Branch 0.8%
- Other instructions 0.0%

* 93.8% memory access instructions, 1.7% implicit memory accesses in other instructions, also counted in their categories

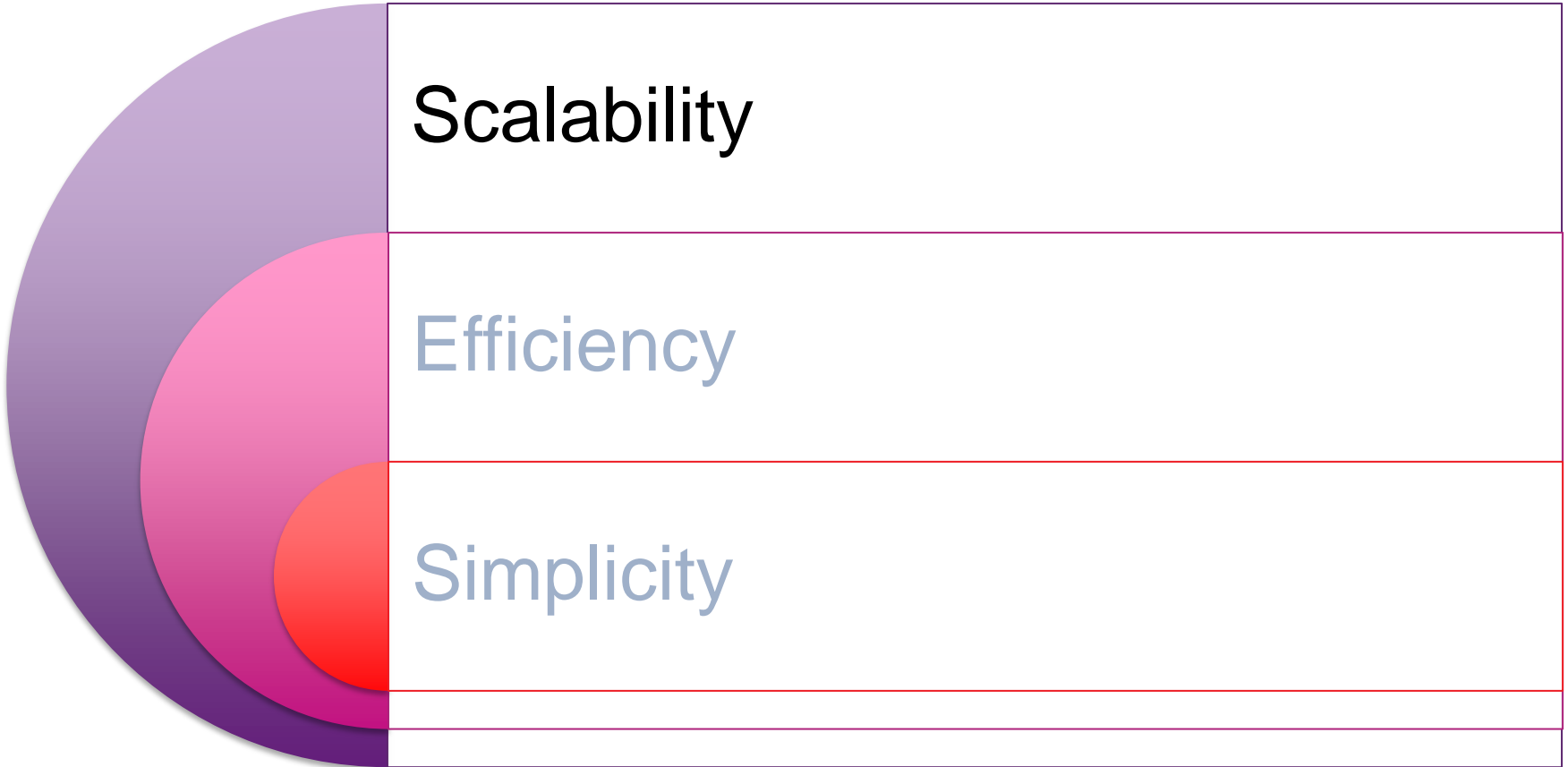
Input/Output Project Files Main Thread Stacks Functions

Main Thread Stacks

Total core time	MPI	Function(s) on line	Source	Position
84.1%		main	{	mmult1.c:73
<0.1%		mmult	mmult(size, nproc, mat_a, mat_b, mat_c);	mmult1.c:156
			res += A[i*size+k]*B[k*size+j];	mmult1.c:63
<0.1%		1 other		
11.4%	11.3%	MPI_Finalize	MPI_Finalize();	mmult1.c:185
1.3%		mwrite	mwrite(size, mat_c, filename);	mmult1.c:177
1.1%	1.1%	MPI_Send	MPI_Send(&mat_c[0], size, MPI_DOUBLE, 0, MPI_COMM_WORLD, MPI_	mmult1.c:171

Showing data from 8,000 samples taken over 8 processes (1000 per process) Allinea Forge 6.0.1 Main Thread View

Reducing trapped capacity on next-gen processors



Towards more vertical & horizontal scalability

INTEL KNIGHTS LANDING



NVIDIA GPUS



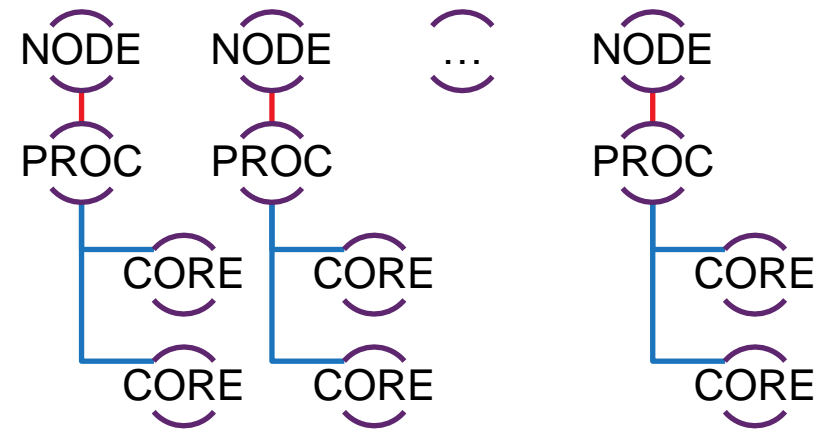
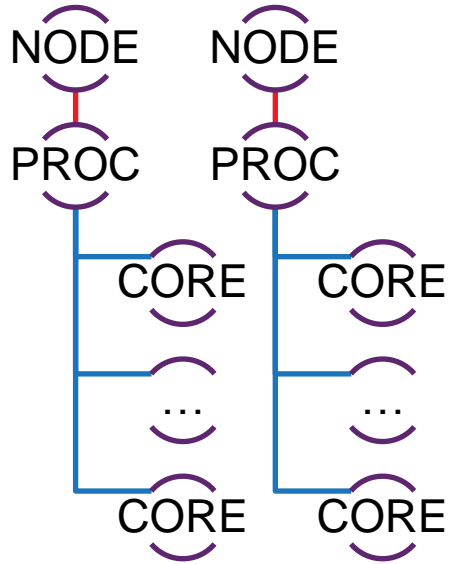
FPGAs



NEXT-GEN INTEL XEON



ARM v8



Towards more vertical & horizontal scalability

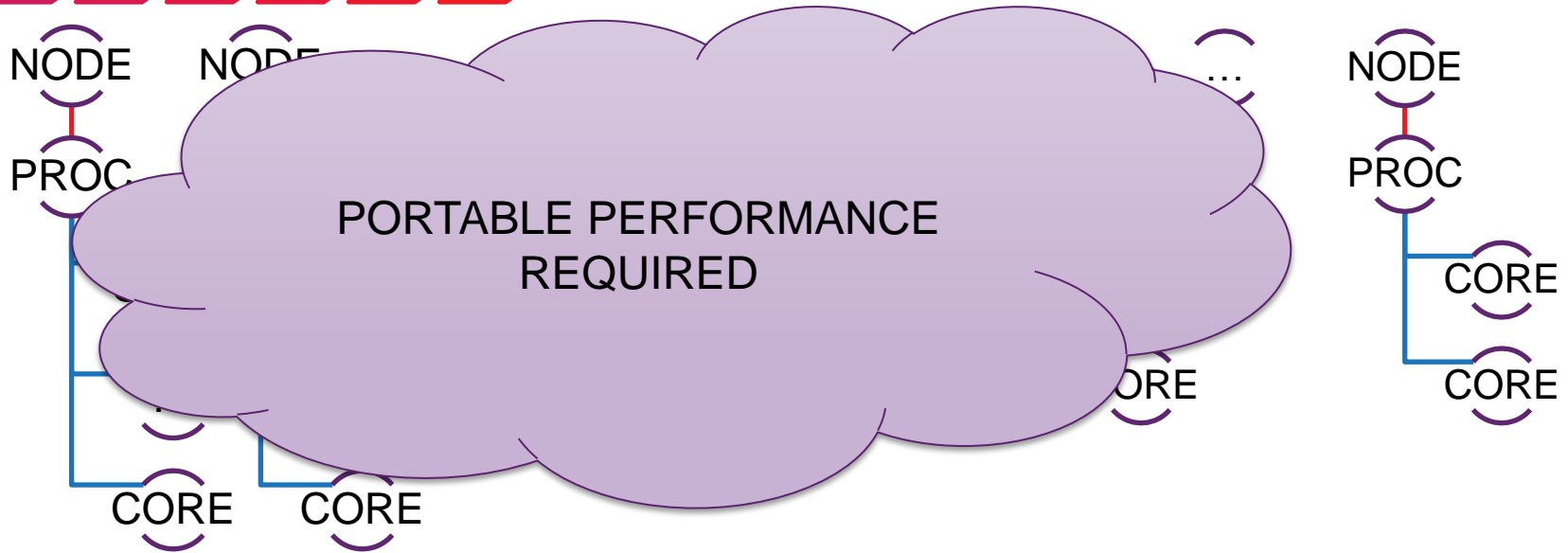
INTEL KNIGHTS LANDING

NVIDIA GPUS

FPGAs

NEXT-GEN INTEL XEON

ARM v8



Support for next generation systems

- **ARMv8 Support**

- Allinea Forge already available
- Allinea Performance Reports for ARMv8 scheduled (H2020 ExaNest)

- **Nvidia GPUs Support (currently supported: CUDA 7.5)**

- CUDA 8.0 expected around GTC 2016

- **Intel KNL Support**

- Will be supported by Allinea at release

What is coming next? (more under NDA)

Allinea DDT

Advanced memory debugging for Intel Knight's Landing
New HTML crash and leak reports
Export debugging data to continuous integration tools (Jenkins, Bamboo etc)

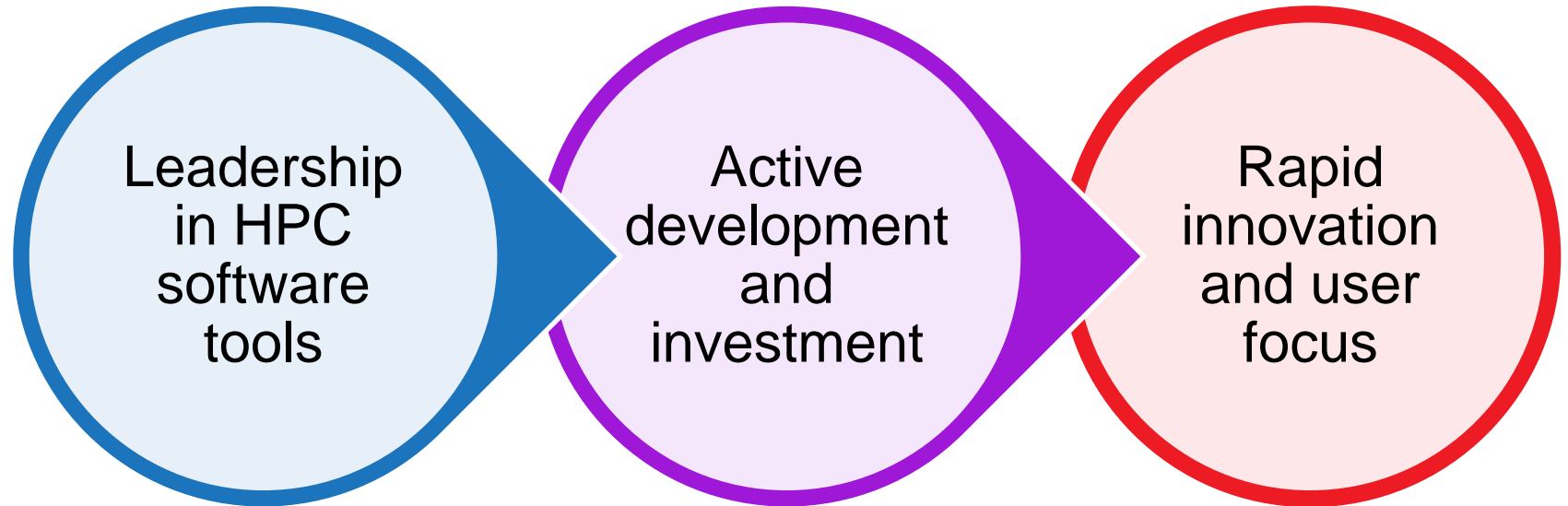
Allinea MAP

Profile selected ranks only
Toggle view between absolute times & percentages
Export function-level performance to continuous integration tools

Allinea Performance Reports

Profile selected ranks only
Toggle view between absolute times & percentages
Add custom section to your own reports
Export metrics to continuous integration tools

Summary: Alinea provides...



- The only scalable tool suite for HPC

- Innovative, extremely scalable tools with unique visual insight

- Providing what science needs, first.

allinea

High performance tools to debug, profile, and analyze your applications

Thank you !

Technical Support team :

support@allinea.com

Sales team :

sales@allinea.com

