Towards Efficiency Computing with Allinea

29 Feb 2016
IISC, Bengaluru

Florent Lebeau
flebeau@allinea.com

allinea

Agenda

13:00 – 13:15	Introduction to Allinea Tools and Latest Changes
13:45 – 14:45:	Profile and Optimise with Allinea Forge
14:45 – 15:45:	Debug with Allinea Forge
13:15 – 13:45:	Application Efficiency with Allinea Performance Reports
15:45 – 16:45:	Hands-on Session on a real application
16:45 – 17:00:	Wrap-Up and questions

Introduction to Allinea Tools

Allinea : an expanding company

- **HPC tools company since 2002**
  - Leading in HPC software tools market worldwide
  - Global customer base
- **Helping the HPC community design the best applications**
  - Unrivaled productive and easy-to-use development environment…
  - … To help reach the highest level of performance and scalability
- **Helping HPC production make the most of their clusters**
  - Unique solutions to reduce HPC systems operating costs
  - Innovative approach to facilitate cutting-edge challenges resolution

allinea

Need to dive into the code ?

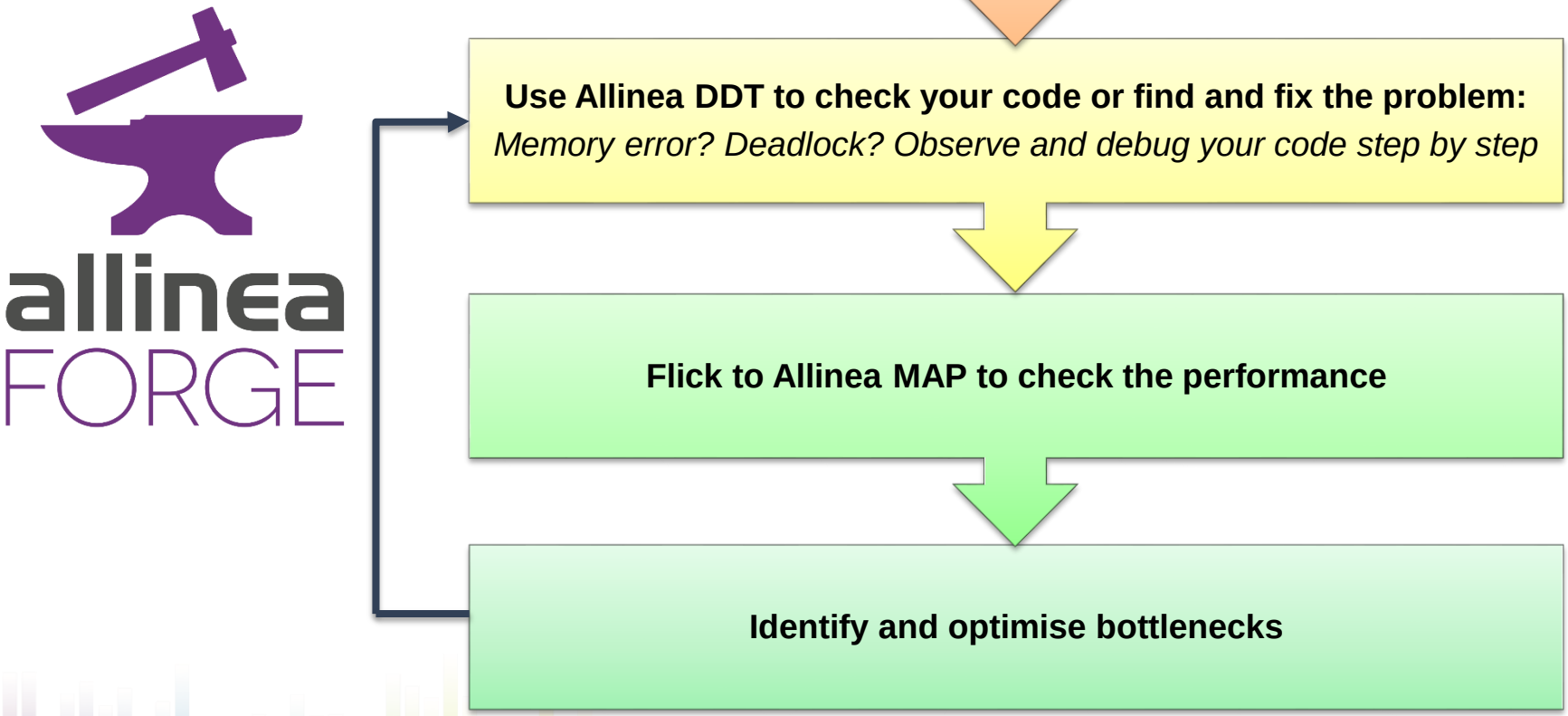
- **Allinea Forge: a modern integrated environment for HPC developers**
  - Rebranding of Allinea Unified (Allinea DDT + Allinea MAP)

- **Supporting the lifecycle of application development and improvement**
  - Productively debug code with Allinea DDT
  - Enhance application performance with Allinea MAP

- **Designed for productivity**
  - Consistent easy to use tools
  - Fewer failed jobs

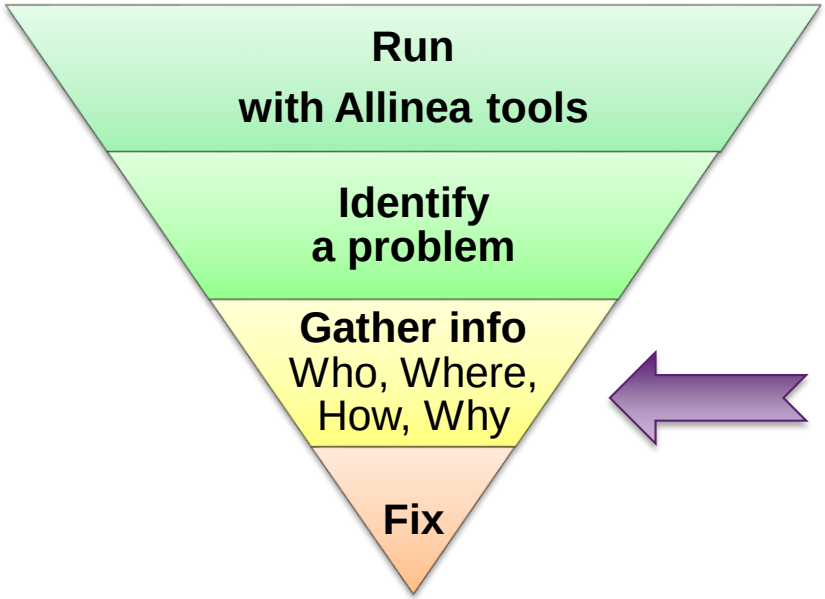
- **Available to you**:
  - Allinea DDT – 2048 processes

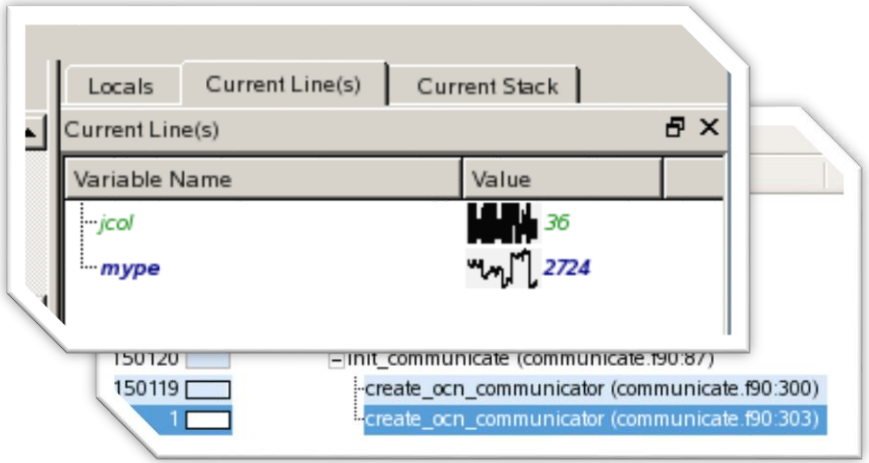
Allinea Forge
One Unified Solution



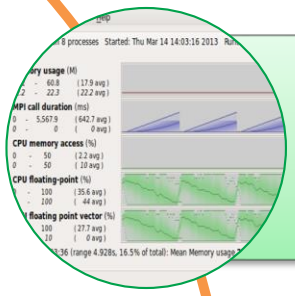
Allinea DDT helps to understand

- ***Who* had a rogue behaviour ?**
  - Merges stacks from processes and threads
- ***Where* did it happen?**
  - Allinea DDT leaps to source automatically
- ***How* did it happen?**
  - Detailed error message given to the user
  - Some faults evident instantly from source
- ***Why* did it happen?**
  - Unique “Smart Highlighting”
  - Sparklines comparing data across processes





Allinea MAP
Performance made easy

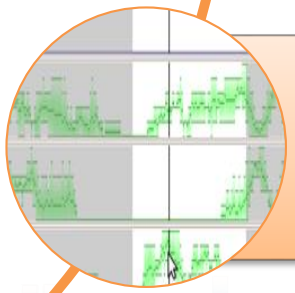


Low overhead measurement

- Accurate, non-intrusive application performance profiling
- Seamless – no recompilation or relinking required

Easy to use

- Source code viewer pinpoints bottleneck locations
- Zoom in to explore iterations, functions and loops



Deep

- Measures CPU, communication, I/O and memory to identify problem causes
- Identifies vectorization and cache performance

Allinea MAP and tracing tools: a great synergy

Simple optimization with Allinea MAP

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

Prepare optimization strategy with Allinea MAP

- Identify loop(s) to instrument
- Identify performance counter(s) to record
- Document performance issues to communicate to profiling experts

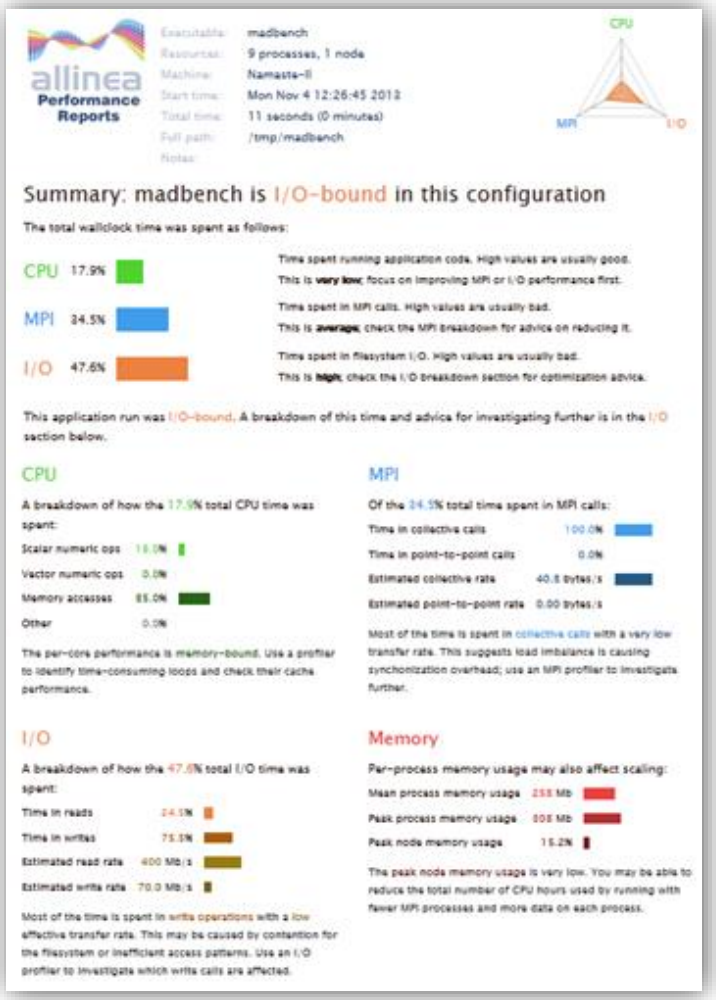
Fine tune the code with tracing tool

- Retrieve low-level details using traces
- Fix up CPU usage to make the code fly

Improve cluster efficiency

- “Optimisation” is not always synonym of “efficiency”
  - Cluster productivity or cluster usage
- Possible efficiency needs during production
  - Define and enforce best practices (scale, parameters…)
  - Provision and validate cluster upgrades and changes
  - Detect & resolve hardware or software faults impacting performance
- Effortless one-touch reports with allinea
  - Generates explicit and readable reports with metrics and explanations
  - Understand optimized HPC applications effortlessly

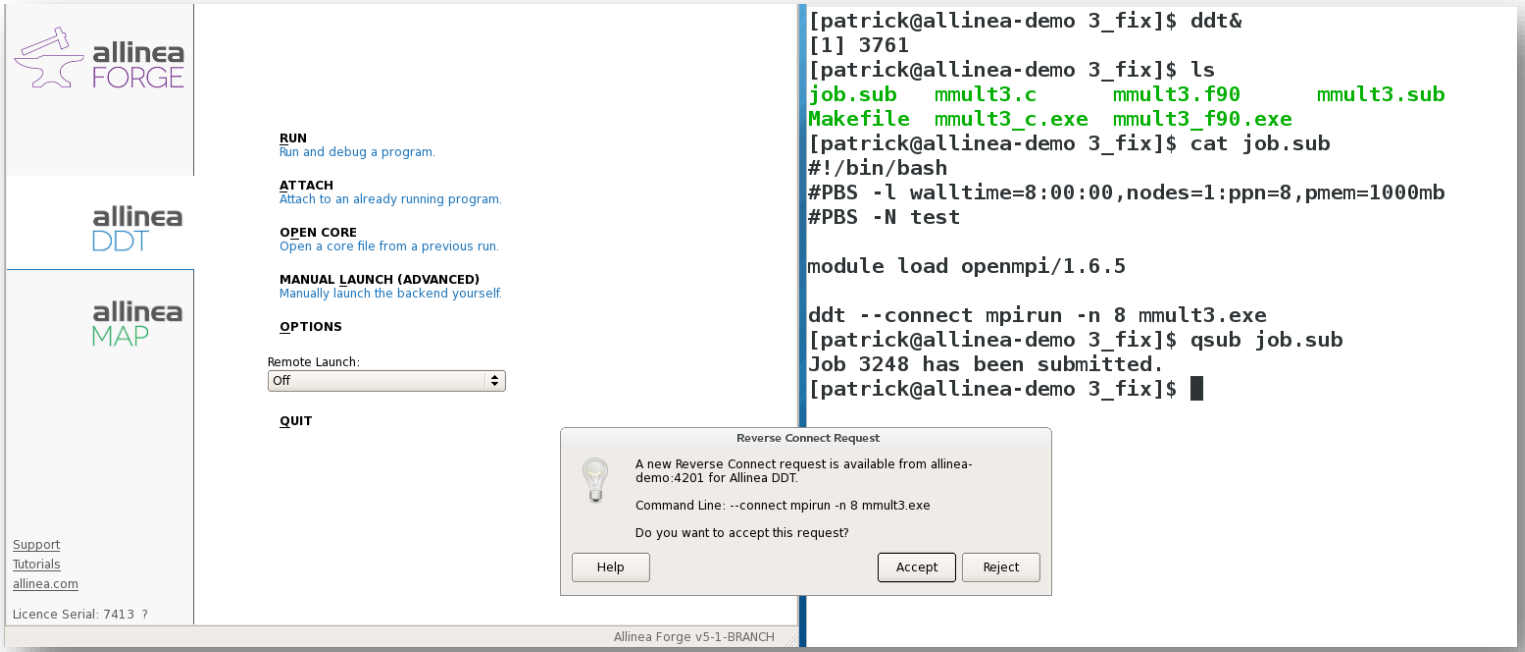
Better runs, quickly



- No source code needed
- Less than 5% runtime overhead
- Fully scalable
- Run regularly – or in regression tests
- Explicit and usable output

Latest Changes

Reverse connect: the end of template files



Profile and Optimise with Allinea Forge

The quest for the Holy Performance

Code optimisation can be time-consuming.

Efficient tools can help you focus on the most important bottlenecks.

How to use Allinea MAP

- Prepare the code

```
$ mpicc –O3 –g myapp.c –o myapp
```

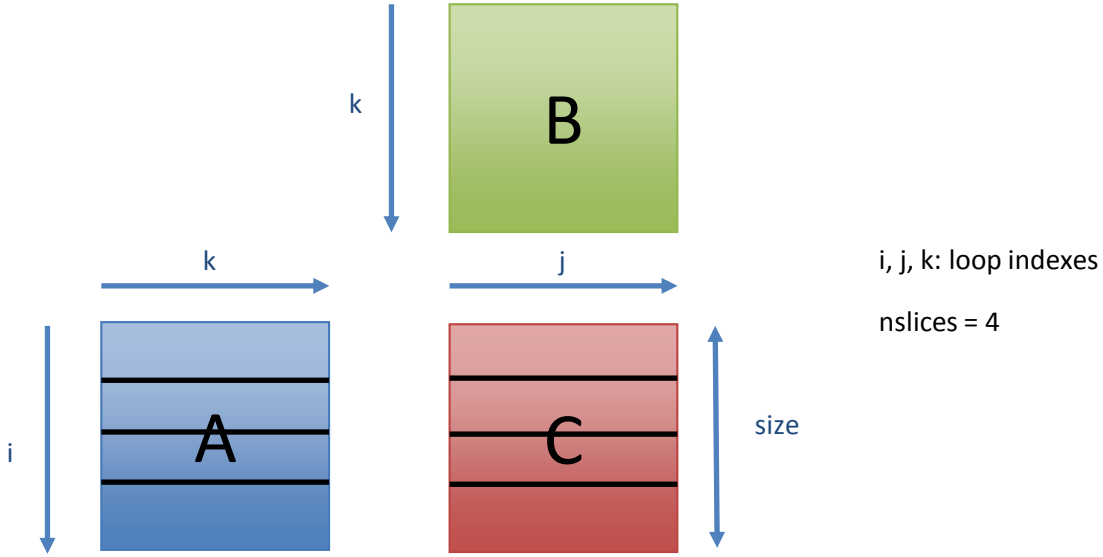
- Profile the application with Allinea MAP

```
$ map --profile mpirun -n 8 ./myapp arg1 arg2
```

- Open the result

```
$ map ./myapp_8p_YYYY-MM-DD_HH-MM.map
```

Tutorial:
Matrix Multiplication: C = A x B + C



Algorithm

1- Master initialises matrices A, B & C

2- Master slices the matrices A & C, sends them to slaves

3- Master and Slaves perform the multiplication

4- Slaves send their results back to Master

5- Master writes the result Matrix C in an output file

Getting started for the workshop

- **Install the Allinea Remote Client**

  Go to : http://www.allinea.com/products/downloads

- **Connect to the cluster with the remote client**

  Open your Remote Client

  Create a new connection:

  Hostname: `training@<IP@ddressProvidedByTrainer>`

  Remote installation directory: `/opt/allinea/forge/`

  Connect!

- **Go to the first exercise**

  ```
  $ ssh –X training@<IP@ddressProvidedByTrainer>
  ```

  Password: allinea

  ```
  $ cd ~/allinea_wshop/1_profiling
  ```

List of IPs for the workshop

#	IP	#	IP
1	52.77.232.241	10	54.254.199.255
2	54.254.222.171	11	54.254.223.97
3	54.254.222.196	12	54.169.68.101
4	54.254.221.247	13	54.254.223.40
5	54.254.218.13	14	54.179.132.63
6	54.169.169.130	15	52.77.214.106
7	54.254.226.37		
8	54.179.164.66		
9	54.254.225.122		

Profiling the application

Go to allinea_wshop/1_profiling

Exercise objectives :

- Compile a code for Allinea MAP
- Profile the application
- Discover Allinea MAP interface and features
- Optimize a simple code

Content

- Source code
- Makefile

What is the bottleneck of the application? How can you optimise it?

Resolving Bugs with Allinea Forge

Debugging by Discipline

Debugging a problem is much easier when you can :

- Make and undo changes fearlessly
  - Use a **source control** (CVS, …)
- Track what you've tried so far
  - Write **logbooks**
- Reproduce bugs with a single command
  - Create and use **test script**

```
$ mkdir logs
$ vim logs/segfault-at-4096-procs

When running lu.E.4096 with the trace-4410.dat set,
the job exited with: "An error occurred in MPI_Send
[li346-209:25319] on communicator MPI_COMM_WORLD
MPI_ERR_RANK: invalid rank".

To reproduce: mpiexec -n 4096 lu.W.4096 trace-4410.dat
on supermuc. Seems to happen every time.

* Tried reading core file with gdb, "File truncated"
* Set ulimit -c unlimited and ran again: ...
```

```
$ logs/segfault-at-4096-procs.sh
Sep 27 15:29: Queued as job.43214
Sep 27 18:01: Running...
Sep 27 19:29: FAIL
```

Debugging by Magic

Any technology sufficiently advanced is indistinguishable from magic.

Unpredictable, dangerous, irresistible.

Learn your spells

Debugging a problem is much easier if you know debuggers

- Prepare the code

```
$ mpicc -O0 -g myapp.c -o myapp
```

- Start Allinea DDT in interactive mode

```
$ ddt mpirun -n 8 ./myapp arg1 arg2
```

- Start Allinea DDT in offline mode

```
$ ddt --offline report.html mpirun -n 8 ./myapp arg1 arg2
```

- Use reverse connect

```
$ ddt --connect mpirun -n 8 ./myapp arg1 arg2
```

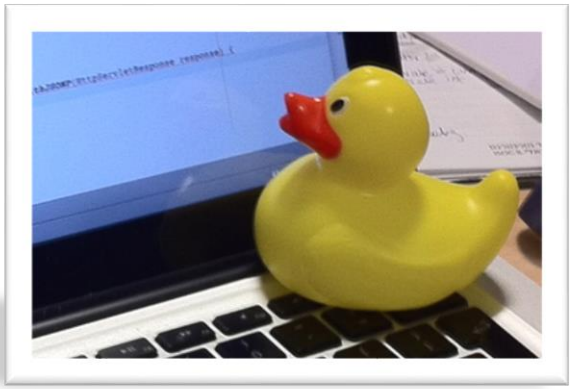
Debugging by Inspiration

Look at the problem, see the solution.

Trust your instincts. Control if they are right.

Debugging by Inspiration

Debugging a problem is much easier if you are inspired :



- Search your inspiration sources
  - Check your **past logbooks**
  - Explain the problem to a **rubber duck**
- Test your instincts
  - **Create tests** (tracepoints, watchpoints, conditional breakpoints…)
- Observe what the debugger is telling you
  - Analyse what the debugger communicates
  - Retrieve information from the debugger (**advanced magic**)

Debugging by Inspiration

- Memory errors can be obvious (segfaults …)
- Sometimes not

- Allinea DDT memory debugging tool enables automatic error detection
  - By activating dmalloc library
  - By adding guard pages
  - On the host as well as on the Xeon Phi

- Different levels of detection brings different debugger behaviour

Exercise 2 : Working on the Optimized code

Go to ~/allinea_wshop/2_debugging

Exercise objectives:

- Compile a code for Allinea DDT
- Discover Allinea DDT interface and features
- Enable memory debugging and debug a memory leak

Content

- Source code
- Makefile

How can you find the memory leak? How can you fix the issue?

Application Efficiency with Allinea Performance Reports

Exercise 3: Analyse the application

Go to ~/allinea_wshop/3_reporting

Exercise objectives:

- Discover Allinea Performance Reports metrics
- Analyse different configurations

Content

- Source code
- Makefile

What is the optimal configuration for the application?

Hands-on session on your application

Exercise 4: Hydro

Go to ~/allinea_wshop/4_hydro

Hydro is a CFD benchmark application developed by the CEA written in C and using MPI and OpenMP.

- https://github.com/HydroBench

Content

- Source code and Makefile

  $ cd Src/

  $ make

- Execution script:

  $ cd ../Bin

  $ ./run.sh

Play around with different configurations ! Modify the code!

Summary

- Develop *your* efficiency with Allinea Forge
  - Optimize your code to reach your goals with Allinea MAP
  - Reduce the number of failed jobs with Allinea DDT
- Improve cluster usage with Allinea Performance Reports
  - Squeeze more jobs within a given time frame
  - Increase research by freeing machine time without hardware investment
  - Help application support teams focus on the right issues

Thank you

Your contacts :

- Technical questions? Florent Lebeau - *flebeau@allinea.com*
- Sales: Avtar Cheema - *acheema@allinea.com*

allinea