

# Allinea DDT Training day

Debug parallel codes efficiently and at-scale



www.allinea.com

### Agenda of the day

- 09:00 09:30 : Registration
- 09:30 10:00 : Presentation Allinea environment
- 10:00 11:00 : Getting started
- 11:00 11:30 : Coffee Break
- 11:30 11:45 : GPU Debugging (demo only)
- 11:45 13:00 : Memory debugging
- 13:00 14:00 : Lunch
- **14:00 14:45 : Incorrect results**
- 14:45 15:30 : Memory leaks
- 15:30 16:00 : Coffee Break
- 16:00 17:00 : Allinea MAP demo, free session bring your own codes !





# Let's talk about us!





### New technologies, more parallelism



## How does my application scale ?



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

#### The total wallclock time was spent as follows:

CPU	17.9%	
MPI	24.3N	
1/0	47.6N	

- Time spent numming application code. High values are usually good. This is very line; focus on impossing MPI or 1; 0 partormance first. Time spent in MPI calls. High values are usually text.
- This is average, check the MPI breakdown for advice on reducing it.
- Time spert in fliesystem I, O, High values are vousily bed. This is **Mark**, check the I-O presidown section for potimization advice.

This application run was 1/0-bound. A breakdown of this time and advice for investigating further is in the 1/0 section below.

#### CPU

A braskdown of how the 17.1% total CPU time was spent:

Scatar numaric ops	11.08	
Vector numeric ops	0.08	
Mamory accesses	81.2%	

The per-core performance is memory-bound. Use a profiler to identify time-consuming wops and check their tachs performance.

0.08

#### 1/0

Other

A breakdown of how the 47.6N total I/O time was



Must of the time is spect in write operations with a low effective transfer rate. This may be caused by contention for the theystem or inefficient access patterns. Use an 1/0 synther to investigate write write cast are effected.

#### MPI



- Time in point-to-point calls 0.0%
- Estimated collective rate 40.8 bytes, 5

Most of the time is spent in converting color with a very time transfer rate. This suggests load imitations is causing synchosization overhead, use an MPI profiler to investigate Author.

#### Memory

Per-process memory usage may also affect scaling:

- Mean process memory usage 218 Mb
- Peak process memory usage 103 Mb

The peak mode memory usage is very low. You may be obta 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

## Need to dive into the code ?

- 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
  - Fewer failed jobs
- Available at EPCC
  - Allinea DDT on 32,768 processes
  - Allinea MAP available as a trial at 8k processes



allinea

allinea

### Debugging in practice The usual method





## Allinea DDT helps to understand

### • Who had a rogue behavior ?

- 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"
    - Coloring differences and changes
  - Sparklines comparing data across processes





# Hands-on workshop

www.allinea.com

### In the beginning was the Word

### 1- Connect to the front-end node :

\$ ssh -Y <username>@login.archer.ac.uk

### 2- Get the training package

\$ cp /path/to/archive/ddt-training.tar.gz ~/

### **3- Setup the environment**

\$ module load ddt

### 4- Read instructions

- \$ evince instructions.pdf
- \$ evince agenda.pdf

### 5- Modify your submission scripts and use aprun to run on Archer !

- \$ vi submission.job
- \$ qsub submission.job

# Getting Started with Allinea DDT 10:00 – 11:00

- Session 1 in training.pdf
- Example 1 : cstartmpi
  - Start an MPI job with Allinea DDT
  - Discovering the GUI
  - Detect and correct straightforward crashes
  - How to replace print statements ?
  - Starting Allinea DDT in offline mode
- Key commands in a nutshell :
  - \$ vi submission\_script.pbs
  - \$ aprun -n 4 ./cstartmpi.exe arg1 arg2
  - \$ ddt -n 4 ./cstartmpi.exe arg1 arg2



### Memory Debugging with Allinea DDT 11:45 – 13:00

- Session 2 in training.pdf
- Example 2 : trisol
  - Enable memory debugging
  - Enable guard pages
  - Resolve straightforward memory-related crashes
- Key commands in a nutshell :

  - \$ aprun -n 4 ./trisol.exe
  - \$ ddt -n 4 ./trisol.exe



### Incorrect Results with Allinea DDT 14:00 – 14:45

- Session 3 on training.pdf
- Example 3 : array
  - Visualize arrays in a distributed environment
  - Filter and detect rogue values in parallel
  - Setup watchpoints
- Key commands in a nutshell :
  - \$ aprun -n 8 ./array
  - \$ ddt -n 8 ./array



# Memory leaks with Allinea DDT 14:45 – 15:30

- Session 4 on training.pdf
- Example 4 : mandel
  - Analyze the current memory usage
  - Analyze memory allocations and statistics
- Key commands in a nutshell :
  - \$ vi Makefile # you need to enable memory debugging !
  - \$ aprun -n 8 ./mandel
  - \$ ddt -n 8 ./mandel







- Allinea MAP demonstration !
- Bring your own code !
- Try more of the exercises (optional exercises) !





# Thank you

Your contacts :

- Technical Support team :
- Sales team :

support@allinea.com sales@allinea.com

www.allinea.com