Welcome!

Virtual tutorial starts at 15:00 BST
eCSE: Supporting Data

ARCHER Virtual Tutorial, Wed 3rd September 2014
Lorna Smith, Chris Johnson, Mark Filipiak, Xu Guo
EPCC
Reusing this material

This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License. 

http://creativecommons.org/licenses/by-nc-sa/4.0/deed.en_US

This means you are free to copy and redistribute the material and adapt and build on the material under the following terms: You must give appropriate credit, provide a link to the license and indicate if changes were made. If you adapt or build on the material you must distribute your work under the same license as the original.

Note that this presentation contains images owned by others. Please seek their permission before reusing these images.
Introduction

• Programme provides funding to ARCHER user community to develop software in a sustainable manner for ARCHER

• Objectives
  • To sustain key codes for the UK computational science community
  • To facilitate efficient use of ARCHER resources through enhanced code performance/functionality
  • To offer a not-for-profit service that provides value for money to the HPC user community and beyond

• Also
  • Develop and sustain codes and communities from new areas
  • Support and encourage early career researchers
Submission Format

• After calls opens, proposals should be submitted via SAFE:
  • [https://www.archer.ac.uk/safe/](https://www.archer.ac.uk/safe/)
  • Please register first if you are not a registered user in SAFE

• Two components to the submission
  • Project Information
  • Project Proposal

• Project Information
  • Mandatory information such as names, proposed start date, travel requested
  • Required resources
    • Primarily for the eCSE team to determine if any additional support required
    • Additional AU’s must however be justified
Submission: Project Proposal Template

- Project Objectives
- Project Overview
- Applicants’ Track Record
- Technical Information
- Computational Benefits
- Scientific Benefits
- Benefits for the ARCHER Community
- Sustainability / Pathways to Impact
- Embedded CSE Support Requested / Work Plan
Project Objectives

• Form part of the proposal assessment criteria
• And if accepted will be asked to report against these objectives
  • Used to assess the final success of your project
  • Should therefore be specific and measurable
• Examples include but are not limited to:
  • The enablement of the scientific community to perform novel and previously untenable simulations
  • A quantifiable improvement in performance or scaling of a code
  • The integration of new algorithms/functionality into a code
  • Measurable outcomes leading to wider accessibility in the user community
  • Project outcomes of specific importance to the ARCHER community
Technical Background

• Demonstrate a good knowledge and understanding of previous and current work in the related area

• May include but is not limited to:
  • A brief summary of the previous / current use of the code
    • HPC platforms used, the software environments for the code running, the number of cores and problem size used, etc
  • The previous / current code performance, scaling and profiling
  • The major algorithms and functional updates related to the code to be used in the proposed project
  • The important prerequisites for the proposed project, e.g. the key algorithms, libraries, software to be installed, etc
Previous code performance, scaling and profiling

- Should allow the panel to understand the current performance of the code on ARCHER
- Ideally results will be on ARCHER, but if not, should address architecture differences
  - Provide confidence results are transferrable
- Need not be your “own” results, but must provide confidence in their accuracy
- Must give confidence that the results are representative of the problems you wish to consider in your proposal
  - i.e. scientific beneficiary systems
  - Need not be same systems but should be representative
Previous code performance, scaling and profiling

• Should demonstrate the codes appropriateness / ability to utilise ARCHER
  • Some codes are more suited to other forms of funding

• Should address current code limitations and motivate developments proposed
  • Profiling evidence
  • e.g. why does scaling tail-off?
  • e.g. how can this be addressed?
  • e.g. can you quantify the expected performance improvements?

• Can be used to provide confidence that the project objectives are realistic and achievable
Previous code performance, scaling and profiling

• The major algorithms and functional updates related to the code to be used in the proposed project
  • Motivated by your performance data

• The important prerequisites for the proposed project, e.g. the key algorithms, libraries, software to be installed, etc
  • Provide confidence that the work can actually be done on ARCHER
    • Particularly important if code has not been run on ARCHER before
  • Helps the eCSE team understand project and support requirements
How do I generate this data?

• The centralised eCSE team can help
  • Either through advise or carrying out some initial benchmarking/profiling
• You can apply for “EPSRC Instant Access”
  • Provides pump priming time for new users
  • Limited number of AUs available over 6 months for testing
• Various tools available on ARCHER to obtain this information
  • Next part of the tutorial discusses this in more detail
Performance data

- **Total wall clock time**
  - System commands (e.g. time) or batch system statistics
  - Built-in timers in the program (e.g. MPI_Wtime)

- **Built-in timers** can be used to get fine-grained timings, e.g., excluding initialization time, or I/O time.
  - No information about hardware related issues e.g. cache utilization
  - Information about load imbalance and communication statistics is difficult to obtain
Performance analysis tools

- On Archer
  - Cray performance tools
    - Works with all compilers
    - For Cray systems only
  - Scalasca
    - Currently works with the Cray compiler only
    - Used on many other systems
Cray Performance Tools

- **Instrument** the code
  - Adds special measurement code to binary

- **Collect** data from a run of the instrumented binary
  - Sampling (statistical averages, low overhead) vs. tracing (data from every traced call, high overhead, lots of data)
  - Guided tracing: trace functions that are not too small and contribute a lot to application’s run time. Cray Automatic Profiling Analysis does this.

- **Analyze**
  - Text based analysis reports
  - Visualization
Steps to Collecting Performance Data

- Access performance tools software
  
  ```bash
  % module load perftools
  ```

- Build application keeping .o files

  ```bash
  % make clean
  % make
  ```

- Instrument application for automatic profiling analysis

  You will get an instrumented program `<name>+pat`

  ```bash
  % pat_build -O apa a.out
  ```

- Run application (in a qsub script)

  You will get a performance file ("<sdatafile>.xf") or multiple files in a directory `<sdatadir>`

  ```bash
  % aprun ... a.out+pat
  ```
Steps to Collecting Performance Data (2)

- Generate text report and an .apa instrumentation file
  \[\texttt{% pat\_report -o my\_sampling\_report [<sdatafile>.xf | <sdatadir>]}\]
- Inspect .apa file
- View sampling report as text or with Cray Apprentice
  \[\texttt{% app2 <sdatafile>.ap2}\]
- Verify if additional instrumentation is needed
APA File Example

# You can edit this file, if desired, and use it
# to reinstrument the program for tracing like this:
#
#    pat_build -O cfd+pat+780378-3005s.apa
#
# These suggested trace options are based on data from:
#
#    cfd+pat+780378-3005s.ap2
#
# Collect the default HWPC group.
#    -Drtenv=PAT_RT_PERFCTR=default
#
# Libraries to trace.
#    -g mpi
#
# User-defined functions to trace, sorted by % of samples.
#    The way these functions are filtered can be controlled with
#    pat_report options (values used for this file are shown):
#    -s apa_max_count=200    No more than 200 functions are listed.
#    -s apa_min_size=800     Commented out if text size < 800 bytes.
#    -s apa_min_pct=1       Commented out if it had < 1% of samples.
#    -s apa_max_cum_pct=90   Commented out after cumulative 90%.
#    Local functions are listed for completeness, but cannot be traced.
#    -w # Enable tracing of user-defined functions.
#    # Note: -u should NOT be specified as an additional option.
#
# 67.53% 6633  bytes
#    -T cfd_
#
# New instrumented program.
#    /fs3/y02/y02/ted/training/201312-CSE-EPCC/reggrid/cfd # Original program.

Effectively a series of command line arguments to pat_build
Generating Event Traced Profile from APA

- Instrument application for further analysis (a.out+apa)
  
  % pat_build -O <apafilename>.apa

- Run application (in a qsub script)
  
  % aprun ... a.out+apa

- Generate text report and visualization file (.ap2)
  
  % pat_report -o my_text_report.txt [<datafile>.xf | <datadir>]

- View report as text or with Cray Apprentice
  
  % app2 <datafilename>.ap2
Using pat_report

- Always need to run pat_report at least once to perform data conversion
  - Combines information from the raw performance data in the xf file (optimized for writing to disk) and the binary to produce an ap2 file (optimized for visualization analysis)
- Generates a text report of performance results
  - Data laid out in tables
  - Many options for sorting, slicing or dicing data in the tables.
    - `pat_report -O <table option> *.ap2`
    - `pat_report -O help` (list of available profiles)
  - Volume and type of information depends upon sampling vs. tracing.
Table 1: Profile by Function

<table>
<thead>
<tr>
<th>Samp%</th>
<th>Samp</th>
<th>Imb. Samp</th>
<th>Imb. Samp%</th>
<th>Group</th>
</tr>
</thead>
<tbody>
<tr>
<td>100.0%</td>
<td>7607.1</td>
<td>--</td>
<td>--</td>
<td>Total</td>
</tr>
<tr>
<td>67.6%</td>
<td>5139.8</td>
<td>--</td>
<td>--</td>
<td>USER</td>
</tr>
<tr>
<td>67.5%</td>
<td>5136.8</td>
<td>1076.2</td>
<td>17.9%</td>
<td>cfd_</td>
</tr>
<tr>
<td>31.8%</td>
<td>2421.7</td>
<td>--</td>
<td>--</td>
<td>MPI</td>
</tr>
<tr>
<td>13.7%</td>
<td>1038.5</td>
<td>315.5</td>
<td>24.1%</td>
<td>MPI_SSEND</td>
</tr>
<tr>
<td>7.2%</td>
<td>547.1</td>
<td>3554.9</td>
<td>89.5%</td>
<td>mpi_recv</td>
</tr>
<tr>
<td>7.1%</td>
<td>540.4</td>
<td>3559.6</td>
<td>89.6%</td>
<td>MPI_WAIT</td>
</tr>
<tr>
<td>3.8%</td>
<td>290.8</td>
<td>319.2</td>
<td>54.0%</td>
<td>mpi_final</td>
</tr>
</tbody>
</table>

Observations and suggestions

MPI Grid Detection:

A linear pattern was detected in MPI sent message traffic.
For table of sent message counts, use -O mpi_dest_counts.
For table of sent message bytes, use -O mpi_dest_bytes.
## pat_report: Hardware Performance Counters

<table>
<thead>
<tr>
<th>Counter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>PERF_COUNT_HW_CACHE_L1D:ACCESS</td>
<td>99236829284</td>
</tr>
<tr>
<td>PERF_COUNT_HW_CACHE_L1D:PREFETCH</td>
<td>1395603690</td>
</tr>
<tr>
<td>PERF_COUNT_HW_CACHE_L1D:MISS</td>
<td>5235958322</td>
</tr>
<tr>
<td>CPU_CLK_UNHALTED:THREAD_P</td>
<td>229602167200</td>
</tr>
<tr>
<td>CPU_CLK_UNHALTED:REF_P</td>
<td>7533538184</td>
</tr>
<tr>
<td>DTLB_LOAD_MISSES:MISS_CAUSES_A_WALK</td>
<td>29102852</td>
</tr>
<tr>
<td>DTLB_STORE_MISSES:MISS_CAUSES_A_WALK</td>
<td>6702254</td>
</tr>
<tr>
<td>L2_RQSTS:ALL_DEMAND_DATA_RD</td>
<td>3448321934</td>
</tr>
<tr>
<td>L2_RQSTS:DEMAND_DATA_RD_HIT</td>
<td>3019403605</td>
</tr>
<tr>
<td>User time (approx)</td>
<td>76.128 secs 205620987829 cycles</td>
</tr>
<tr>
<td>CPU_CLK</td>
<td>3.048GHz</td>
</tr>
<tr>
<td>TLB utilization</td>
<td>2956.80 refs/miss 5.775 avg uses</td>
</tr>
<tr>
<td>D1 cache hit,miss ratios</td>
<td>95.1% hits 4.9% misses</td>
</tr>
<tr>
<td>D1 cache utilization (misses)</td>
<td>20.22 refs/miss 2.527 avg hits</td>
</tr>
<tr>
<td>D2 cache hit,miss ratio</td>
<td>91.8% hits 8.2% misses</td>
</tr>
<tr>
<td>D1+D2 cache hit,miss ratio</td>
<td>99.6% hits 0.4% misses</td>
</tr>
<tr>
<td>D1+D2 cache utilization</td>
<td>246.83 refs/miss 30.853 avg hits</td>
</tr>
<tr>
<td>D2 to D1 bandwidth</td>
<td>2764.681MB/sec 220692603786 bytes</td>
</tr>
</tbody>
</table>
perftools documentation

% module load perftools
% man intro_crarpat
% man pat_build
% man pat_report
Relevant Information

- After calls opens, proposals should be submitted via SAFE:
  - [https://www.archer.ac.uk/safe/](https://www.archer.ac.uk/safe/)
  - Please register first if you are not a registered user in SAFE
- Information and guidelines for applying can be found at:
  - [https://www.archer.ac.uk/community/eCSE/eCSE_ApplicationGuidance.pdf](https://www.archer.ac.uk/community/eCSE/eCSE_ApplicationGuidance.pdf)
  - [https://www.archer.ac.uk/community/eCSE/eCSE_ProposalTemplate.doc](https://www.archer.ac.uk/community/eCSE/eCSE_ProposalTemplate.doc)
- Applicants can request guidance from the centralised CSE team before submission:
  - Please contact ARCHER helpdesk: support@archer.ac.uk
Goodbye!

Virtual tutorial has finished