BENCHMARKS FOR SYSTEM PROCUREMENTS
METHODS & SCOPE AT JSC

26 September 2023 | Andreas Herten | Forschungszentrum Jülich, Jülich Supercomputing Centre

Member of the Helmholtz Association

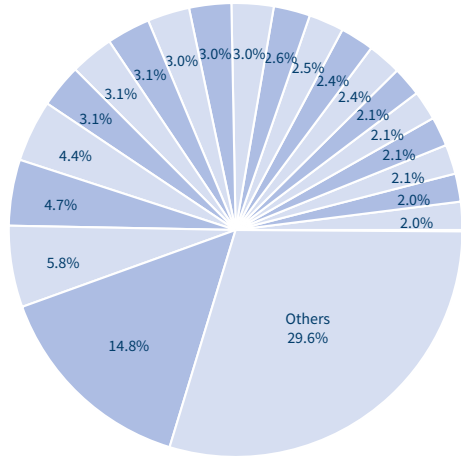
JÜLICH
Forschungszentrum

Outline

- Motivation
- Benchmarks
  - Application Benchmarks
  - Synthetic Benchmarks
  - Rules
- Technical Setup
- Next Steps
- Lessons Learned
- JUPITER

JSC Workload

- JSC: HPC resources for Forschungszentrum campus, state (NRW), Germany, Europe
- Entirely scientists/researchers
- Compute time through peer-review
- Very heterogeneous workload
- Physics, climate, biology, chemistry, AI, …
- Support structures
  - SDLs *Simulation & Data Labs*: Domain-centric labs interfacing between HPC and sciences
  - ATMLs *Algorithms, Tools, & Method Labs*: HPC-centric lab applying methods/tools/hardware/… domain-agnostically



GPU hours

JÜLICH Forschungszentrum

Task:

Respect current & anticipated workload in procurements of new systems

Benchmark Suite Overview

- Main assessment: **Total Cost of Ownership**
  - Proposals ranked by *workload intensity* (how much workload over system lifespan)
  - Also energy consumption respected (simplified)
  - Master formula which calculates value for ranking
  - Metric: runtime of applications
  - Applications-based

Benchmark Suite Overview

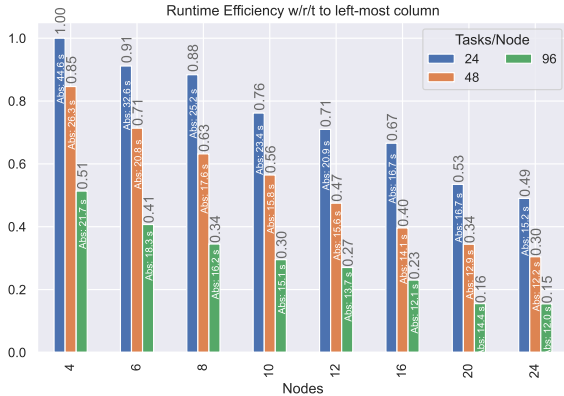
- Main assessment: **Total Cost of Ownership**
  - Proposals ranked by *workload intensity* (how much workload over system lifespan)
  - Also energy consumption respected (simplified)
  - Master formula which calculates value for ranking
  - Metric: runtime of applications
  - Applications-based
- New: High-Scaling Benchmarks
  - For Exascale procurement
  - Respect large-scaleness of system
  - Run dedicated workload on entire system
  - Metric: scaling efficiency
  - Applications-based

Benchmark Suite Overview

- Main assessment: **Total Cost of Ownership**
  - Proposals ranked by *workload intensity* (how much workload over system lifespan)
  - Also energy consumption respected (simplified)
  - Master formula which calculates value for ranking
  - Metric: runtime of applications
  - Applications-based
- New: **High-Scaling Benchmarks**
  - For Exascale procurement
  - Respect large-scaleness of system
  - Run dedicated workload on entire system
  - Metric: scaling efficiency
  - Applications-based
- Also: **Synthetic benchmarks**

Application Benchmarks Introduction

- Central repository at JSC since 2017
  - 50 benchmarks defined
  - Various quality
  - Mainly created for procurements, but also other
  - Highest quality usually for procurements, but some outdated
- Well-defined, simplified versions of full-scale applications running on JSC systems
- Used for TCO, High-Scale



Example plot of TVB-HPC (not in any current procurement)

JÜLICH Forschungszentrum

Application Benchmarks Creation

Selection

1. Analysis of system workload
2. Discussions with SDLs/ATMLs, scientists
3. Analysis of reference applications
4. Suitability
   - Simplification
   - Input data
   - Robustness
   - Verifiability
   - …
5. Balance of full set (domains, programming models, profiles, …)

Application Benchmarks Creation

Selection

1. Analysis of system workload
2. Discussions with SDLs/ATMLs, scientists
3. Analysis of reference applications
4. Suitability
   - Simplification
   - Input data
   - Robustness
   - Verifiability
   - …
5. Balance of full set (domains, programming models, profiles, …)

Application Benchmarks Creation

Selection

1. Analysis of system workload
2. Discussions with SDLs/ATMLs, scientists
3. Analysis of reference applications
4. Suitability
   - Simplification
   - Input data
   - Robustness
   - Verifiability
   - …
5. Balance of full set (domains, programming models, profiles, …)

Creation

- Define benchmark case, execution size
- Shrink/simplify
- *Convert to reference metric (usually: rate → time)*
- Add verification
- Create analysis, profile, scaling plots
- Hardening, cross-review

High-Scaling Benchmarks

- Idea: Execute on full* JUWELS Booster (*: 50 PFLOP/s peak sub-partition), compare to full* JUPITER Booster (*: 1000 PFLOP/s peak sub-partition)
- Weakly scale workload 20×
- Provide 3 variants: small, medium, large memory usage
- Hard task, with many node-h invested; looking into a crystal ball

JUPITER Application Benchmarks

- JUPITER: Largest procurement to date
- >18 months of work
- >30 people involved
- 1(-3) associated people (*captains*) per benchmark
- Meetings every two weeks
- Gitlab issue tracker, status tracker (11 points)
- 16 TCO application benchmarks
- 5 High-Scaling application benchmarks (from TCO apps)
- Also: 1 modular Cluster+Booster benchmark

JUPITER Application Benchmarks

- JUPITER: Largest procurement to date
- >18 months of work
- >30 people involved
- 1(-3) associated people (*captains*) per benchmark
- Meetings every two weeks
- Gitlab issue tracker, status tracker (11 points)
- 16 TCO application benchmarks
- 5 High-Scaling application benchmarks (from TCO apps)
- Also: 1 modular Cluster+Booster benchmark

JUPITER Application Benchmarks

- JUPITER: Largest procurement to date
- >18 months of work
- >30 people involved
- 1(-3) associated people (*captains*) per benchmark
- Meetings every two weeks
- Gitlab issue tracker, status tracker (11 points)
- 16 TCO application benchmarks
- 5 High-Scaling application benchmarks (from TCO apps)
- Also: 1 modular Cluster+Booster benchmark

1. Source code available
2. Input data available
4. JUBE integration
11. Description, documentation

JUPITER Application Benchmarks

- JUPITER: Largest procurement to date
- >18 months of work
- >30 people involved
- 1(-3) associated people (*captains*) per benchmark
- Meetings every two weeks
- Gitlab issue tracker, status tracker (11 points)
- 16 TCO application benchmarks
- 5 High-Scaling application benchmarks (from TCO apps)
- Also: 1 modular Cluster+Booster benchmark

1. Source code available
2. Input data available
4. JUBE integration
11. Description, documentation

- Amber
- Arbor
- Chroma
- GROMACS
- ICON
- JUQCS
- nekRS
- ParFlow
- PIConGPU
- Quantum ESPRESSO
- SOMA
- MMoCLIP
- NLP
- ResNet
- DynQCD (Cluster)
- NAStJA (Cluster)

JÜLICH Forschungszentrum

Synthetic Benchmarks

- Selected *artificial* benchmarks
- Test specific design parameters/features of system
  - Network connectivity, performance; MPI
  - I/O
  - Compute
  - Memory
- Evaluated separately, relatively

Rules

- Benchmarks executed as per recipe/rules given
- Fix version of application
- Response with execution time, strong scaling, log files
- Modifications
  - ✗ Workload, algorithm, precision
  - ✓ Dependencies, libraries
  - ✓ Compiler options
  - ✓ Directives
  - ✓ Programming model
- Changes are to be documented

Technical Setup

- One Gitlab repository per benchmark
- Every benchmark structured similarly/identically
  - Each benchmark in JUBE, using `platform.xml` for system-independence
  - Sources as Git submodules (if possible)
  - Internal data, evaluation included
  - Descriptions, rules, caveats in `DESCRIPTION.md`
  - Machine-readable files with metrics
- Tendering package auto-generated from Gitlab
  - Each benchmark archived to tarball (submodules resolved)
  - `DESCRIPTION.md` converted to TeX and assembled to Benchmarks Descriptions document
  - Including checksums for tarballs
- Delivered via website

JUBE Example

```
- name: systemParameter
  init_with: platform.xml
  parameter:
    - name: preprocess
      _: $modules
    - name: executable
      _: myapp
    - name: args_exec
      _: input.json
    - name: queue
      tag: "baseline|scaling_
      _: booster
    - name: queue
      tag: "exa_tiny|exa_smal
      _: largebooster
```

Next Steps

- Despite long preparation period, some things still to do *(there is never enough time)*
- Most important: Continuous Benchmarking
  - All benchmarks align well in structure
  - → Easy to integrate into CI workflow for CB (with Jacamar)
  - Useful for regression testing (system, software, benchmark), monitoring, system bring-up
- After procurement: Implement improvements, publish our work *somehow*

Lessons Learned

- Large-scale systems need large-scale workloads
  → Hard to prepare with production system
- Time-intensive hunt for workload sizes
- Many *traditional* HPC workloads, few AI workloads for benchmarking
- Verification is hard
- Crystal-ball-iness when preparing workloads for machine of unknown size
- Even between corners of our HPC block, things can be lost in translation

JUPITER

The Arrival of Exascale in Europe

fz-juelich.de/jupiter | #exa_jupiter

Funding Agencies:

EuroHPC Joint Undertaking

Federal Ministry of Education and Research

Ministry of Culture and Science of the State of North Rhine-Westphalia