The Worlds Worst Benchmarks

Colin Honess **Robert Klute**
Hewlett-Packard **Hewlett-Packard**

How do people use benchmarks?

- Competitive comparison
- Proof of concept
  - Will this newly-proposed solution work?
  - Will the existing solution scale?
  - How much extra hardware do I need?
- Performance problem troubleshooting
- Justifying a decision or a marketing claim

The *strengths* of industry standard benchmarks

- First-cut competitive comparison
- Indicator of what workload types a machine is good at
- Indication of scalability
- Promote performance innovation
- Published source of tuning advice

The *weaknesses* of industry standard benchmarks

- The solution does not have to be practical – only meet the vendor's goals: fastest, cheapest, ….
- No vendor posts results for every benchmark
- Not necessarily apples-to-apples
- There is lots of pressure to make 'the numbers'
  - Everyone finesses
- It NEVER matches your workload, your application, the hardware you would buy.

Why do your own benchmark?

- How well the system runs your workload
  - Your data
  - Your applications
  - Your transaction mix
  - Your exact hardware and software configuration
- Within parameters you feel are reasonable
  - You define how much tuning is reasonable
  - Your test methodology
  - Your staff heavily involved in the testing
- Your criteria for success
- It's more than just the result…

It's more than just the result…

- How do you like working with this vendor?
  - How committed are they to your success?
  - Do they put resources behind the promises?
  - Are the engineers capable?
  - Are they willing to share their knowledge?
- How do you like working with these products?
  - Do they work?
  - How easy are common tasks?
  - How accessible is the documentation?
- It's an awesome opportunity to learn!
  - Ask lots of questions – *it's free advice!*

Your Data

- Using REAL data?
  - How confidential is it?
  - How are you going to protect it?
  - How is the test center going to protect it?
  - How are you going to collect it?
  - How will you get it there?
  - How will you load it?
  - How long will the load take?
  - How are you going to delete it afterwards?

Your Data

- Using manufactured data?
  - Is it really representative of the real data?
  - How are you going to create it?
  - How long will it take to create?
  - How can you test the creation process?
- Realistic data volumes
  - Unrealistic caching will seriously distort performance
  - Locking issues may arise from too small a dataset
  - Data structures might look very different for different data volumes
- Real-life overflow and fragmentation

Your Application

- How much of your application do you need to test?
  - All/most/some/tiny subset of the functionality?
  - Everything you choose *not to* test is a potential problem
  - Everything you choose *to* test increases complexity
  - Does your system run multiple workloads?
  - *Try to distill down to the significant ~20 percent*
- The worst thing you can do is a micro-benchmark
- Beware of scope creep
- Do you have technical support available?

Your Workload

- Should reflect real-world use of the system
  - Real transactions
  - Real transaction data
  - Real mix
  - Real user counts
- How are you going to generate the workload?
  - At what "layer" in the software stack?
  - Could the workload generator be the bottleneck?
  - Will the workload generator consume much resource?
- Is it repeatable?
  - How close are results from supposedly identical runs?

Your Hardware Configuration

- More than CPUs and disks
- Your network configuration
  - Need remote access?
    - Setting up Firewalls and security takes time and planning
  - Need multiple networks?
  - How many tiers?
  - Is there enough bandwidth?
  - Is there too much?
  - Is latency important?
- Can your application take advantage of the hardware?

Your Software Configuration

- Operating system, application, middleware versions and associated patches
- Storage configuration and data placement
- Parameters tuned as you want them

Your Test Methodology

- What tests will you run? In what order? How many times? Fallback plans and prioritization…
- Support for the scripts during the benchmark – can you get problems fixed/changes made quickly?
- How do you know the benchmark is running properly?
  - Failure can be *very* fast!
- How to reset the environment between tests?
- Consider the people and the time it will take to test:
  - Benchmarks are stressful and consume long hours
    - Don't schedule 24 hour days, or even 12 hour days.
    - Leave weekends for slack and down time.
  - They are one of a kind, every time
- *Murphy WILL pay a visit.*

Your Criteria for Success

- Finding out how fast it will go ALWAYS fails
- You need a fixed goal –
  - "100K transactions per second, in the ratio… with 95% of transactions completing within the following defined response times…"
  - "Batch A to complete in N hours, when run simultaneously with jobs B, C and D."
  - "Sustained backup rate of N GB per hour."
  - "Throughput will increase 75% as 16 CPUs are added, with no more than a 15% degradation in response time of transaction Y."
- Failure is a successful benchmark outcome if it prevents that failure in production!

A good benchmark is…

- Run for the right reasons
- Representative of your environment and workload
- Meticulously planned, with contingencies
- Very hard work
- A fabulous learning opportunity
- An opportunity to evaluate both people and products.