Performance Testing

Performance Testing Results

- Fiz Karlsruhe Fedora Performance and Scalability Wiki
- Fedora Repository 2.2.4 Performance Tests
- Sun Testing

Performance Testing Initiative

This document is the beginning of a new performance testing initiative for the Fedora Repository and components integrated into Fedora Commons Framework. This work builds upon the superb performance tests and analysis performed by Fiz Karlsruhe, Sun Microsystems (now Oracle) and the performance testing included in each release of the Fedora Repository. We hope this will provide a common performance testing framework and begin a community effort to characterize Fedora performance to document best practices for configuration and identify bottlenecks for subsequent software development. It is presumed that most performance testing will be black-box (through the APIs) but instrumentation may be added to help find bottlenecks.

Goals Include:

- Bottleneck analysis - Eliminate or remove bottlenecks in the current code base
- Feature analysis - Guide the addition of new feature with performance information
- Tuning analysis - Provide information to help tune installations and identify if your installation is under performing
- Sizing information - Provide information to permit installation to size their systems and subsystems
- Performance scalability test
- Size scalability test

Testing Framework Requirements

Performance testing will be accomplished using testbeds kindly provided by a number of community sources. Testing can also be accomplished using Cloud infrastructure such as Amazon's offerings. Individual user organization can stand up their own versions of the testing framework for their unique infrastructure. Details of the test configurations will be documented in this Wiki as the configurations are implemented.

In most cases a test configuration will always consists of at least four parts:

1. A load injector
2. An instance of a Fedora Repository
3. A relational database
4. A collection point for the results

Other typical components include:

1. A resource index (RDF database)
2. An OIA-PMH provider
3. A search engine
4. Ingest components
5. Client software
6. Other framework components or services
7. Fedora Repository backend services

A number of frameworks are used by the community including several home-grown test platforms. Two well supported open-source frameworks are Apache JMeter and The Grinder. Either are well suited for this purpose and have their proponents. In many cases tests written for one framework can easily be re-purposed for another. Hopefully, experience with more than one framework and tests from all will be shared here. Just as important is publication of test results and analyses.

Testing Frameworks

- Using The Grinder
- Using Apache JMeter (Call for Participation)
- Bamboo
- Compute and storage infrastructure
- Performance tests (code)
- Reference test data
- Report publishing facility

Test Resources

To be valid, a performance test must minimize or isolate external influences. This can be done with a dedicated platform (and network if a multi-host configuration is used). Unfortunately, cloud platforms are shared and the performance allocated to any given set of processes varies with time. To utilize a ahared platform, you must monitor it and factor its load into the performance calculations. This can be complicated since there are so many potential factors to consider but "rough order of magnitude" results are still quite useful. This is still useful and may be the most practical approach for interim performance test.

- Best Accuracy - Isolated Platform
    - Must be provided by one or more hosts

    - Also good if on isolated LAN segment
    - Dedicated disks particularly helpful and can test various storage subsystems
    - Separate machine for database and/or resource index
    - Separate machine for load injection
    - Costs system admin time to set up but the installation can be reused
    - Likely only one or a small number of configurations can be supported
    - Some costs in moving test data sets around
    - It would be nice to permit a Bamboo agent to kick off tests
    - Reports can be sent to a common location

- Most Convenience - Cloud Platform
    - Costs cash money but only during testing
    - Difficult to eliminate variables so results will not be as accurate
    - Relatively easy to provide a number of configurations (but there is setup time)
    - AMI or equivalent must be built (though some of this can be done automatically)
    - Shared LAN/WAN only choice
    - Shared disk only choice
    - Easy to incorporate Bamboo
    - Reports can be sent to a common location

There can be any combination of the above but it can start simply, built incrementally.

Questions to Answer.

What is the minimal useful starting point?
What is the first reference platform?
What to we consider the first key tests?
Who will write tests?
When (what events) are tests run?
Can we test a branch?
What is the overlap with integration/functional tests?

Starting Configurations

Configuration 1 -

- Single server-grade host with direct attached disks
    - Load injector on server
- Bamboo server from DuraSpace
- Publishing at DuraSpace

Configuration 2 -

- Cloud server set
    - Fedora Host Server Instance and cloud storage (EBS equivalent)
    - Database Server Instance and cloud storage (EBS equivalent) or combined with Fedora Instance
    - Resource Index Server Instance and cloud storage (EBS equivalent) or combined with Fedora Instance
    - Load Injector Instance
- Bamboo from DuraSpace
- Publishing at DuraSpace