Evolving Use of JBoss Clustering

Saul A. Kravitz
Director, Software Engineering
J. Craig Venter Institute

Hype or Reality?

- Leverage J2EE Wherever You Need it!
- JBoss Great for Scientific Computing!
- JBoss's *Clustered* JMS is Powerful!

➔ Reality!

Outline

- DNA Sequencer Integration
- Experience with Clustering
  - ✓ DNA Sequence Analysis
  - ✓ Sequence Similarity Search
- Conclusions

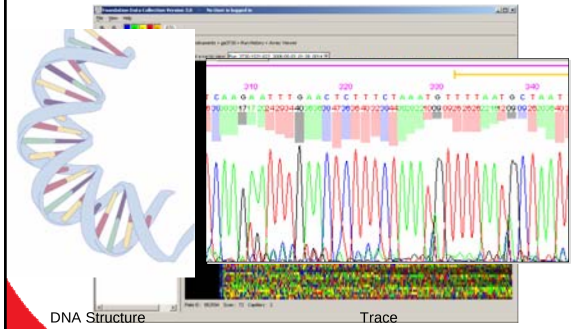
Who REALLY Did the Work

- Pete Davies
- Indresh Singh
- Scott Collins
- Tom Dolafi
- Chris Lemieux
- Sean Murphy
- Adam Resnick
- Angelo Trivelli
- Bryan Yu

J. Craig Venter Institute **Joint Technology Center**

- Established 6/2003
- One of top 5 DNA sequencing centers
- 80M Seq/yr capacity
- 100+ ABI 3730xl sequencers
- 24x7 operation

Processing Sequencing Traces



Initial Problem Definition — 2003

- Support 100-400 DNA sequencers
- Each outputs “runs” of 96 traces
  - ✓ Run generated every 30-120 minutes
  - ✓ 30MB of data/run (~36Gb daily)
- Traces reduced, analyzed, persisted
  - ✓ Near real time requirement
  - ✓ Analysis based on C/C++ executables
  - ✓ Heterogeneous analysis within a run

7

Key Features/Requirements

- Robustness with Intermittent Connectivity
  - ✓ Servers ←→ DBs
  - ✓ Sequencers ←→ Servers
  - ✓ Automatically reestablish connections
  - ✓ Orderly behavior when connections reestablished
- Intranet-Based Laboratory Management
  - ✓ Monitoring & administration of 100+ sequencers
- Easy Deployment to 100+ sequencers
- Scalability
  - ✓ Ability to add additional sequencers
  - ✓ Ability to clear backlog after outages

8

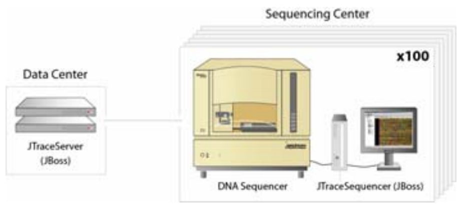
Systems Issues

- Sequencers complete run ~30-120 min
  - ✓ Typically started at roughly the same time
- With 100+ sequencers
  - ✓ Bursts of data
  - ✓ “smoothing” of data inflow required
  - ✓ Occasionally, several days data accumulates
- Constraints
  - ✓ I/O bound - Data transport from sequencers
  - ✓ CPU bound - data processing
  - ✓ I/O bound – data persistence.

9

Solution — JTrace

- Scalable application: 2 JBoss Configs
  - ✓ JTrace Sequencer – on each sequencer
  - ✓ JTrace Server –on cluster in data center



10

JTraceSequencer

- Compact configuration of JBoss
- Integrates with vendor JBoss 2.4 instance via installed EJBs
- Functionality:
  - ✓ Interact with user for run setup
  - ✓ Detection and transfer of generated data
  - ✓ Web/EJB accessible instrument monitoring
  - ✓ HTTP access to sequencer files

11

Detection and Transfer of Data

- Polls instrument directory for new runs
- When run is found:
  1. Jars the run and digitally signs jar
  2. Sends DataAvailable JMS message to Server
  3. Moves Jar to “pickup” area
  4. Server retrieves data via HTTP GET and deletes data with HTTP DELETE after validated transfer
- Sends followup DataAvailable messages
  - ✓ data which has yet to be removed
  - ✓ Provides robustness for network/server outages

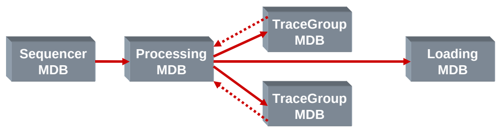
12

Solution Addresses Requirements

- Robustness with Intermittent Connectivity
  - ✓ Servers/Sequencers tolerate intermittent connections
  - ✓ Orderly behavior when all sequencers reestablish connection
  - ➔ JMS / HTTP data transport tolerates outages
  - ➔ JMS connections reconnect after outage
  - ➔ "Pull" data retrieval works at server's pace
- Intranet-Based Laboratory Management
  - ➔ Provided by HTML/EJBinterface on sequencers
- Easy Software Deployment to 100+ sequencers
  - ➔ Hot deploy EAR via shared file system
- Scalability of sequencers and processing capacity
  - ➔ Both sequencers and servers scale trivially

13

JTraceServer



- Retrieves and validates run via HTTP Get
- Deletes run from sequencer via HTTP delete
- Identify homogeneous computations
- Manage fork/join
- N-way Fork
- Join
- Controls execution of [illegible]
- Exploit multi-core architecture and application parallelism
- Persists file and RDB data

14

Fork/Join as JMS Topic vs. Queue

- Fork == JMS Topic?
  - + Topic is point to multipoint
  - – Same subunit sent to >1 subscribers
  - – or, Use message selectors
- Fork == JMS Queue?
  - – Queue is point to point, but
  - + Queue can have >1 MDB instances
  - + Each instance receives one subunit
  - + Load balancing across a cluster

15

Fork/Join using JMS Queue -- Tradeoffs

- Advantages
  - ✓ Retries on failure:
    - Declaratively control JMS based on JMS message redelivery semantics
  - ✓ Concurrency Control within a node
    - Declaratively control within a JBoss instance based on allocation of MDB instances
  - ✓ Scalability:
    - Manage cluster capacity by adding nodes
- Disadvantages
  - ✓ Timing constraints
    - Transaction timeout must be managed

16

Fork Pseudo-code

```
Create msgIDs HashSet;
Create replyQueue;
foreach (message to be sent) {
  Create a message;
  Populate message, with replyQueue;
  Send asynch message;
  Add the message ID to msgIDs;
}
Commit asynch messages;
  // send all or none
```

17

Join Pseudo-code

```
long waitForTime = TRANSACTION_TIMEOUT-10000;

while (replies outstanding, and waitforTime remains ) {
    Wait for replyMessage  on replyQueue for waitForTime;
    waitforTime = remaining wait time;
        if (replyMessage != null) {
               correlate with sent messages;
               processReply(replyMessage);

        }
        if (waitForTime < 0) {
                    //Log timeout error, throw exception
        }
        if (all replies received, but correlation errors) {
                    //Log correlation error, throw exception
        }
}
```

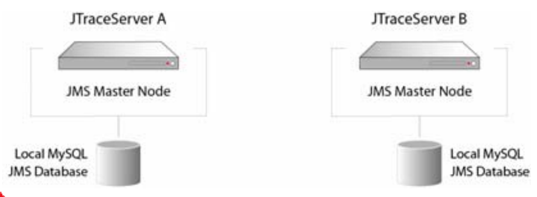
18

JMS Subtleties

- Handle DB Corruption Gracefully
  - ✓ It will happen…
  - ✓ Use appropriate RDBs
- Long-Running Transactions
  - ✓ Work around transaction timeouts
- Send small messages
  - ✓ Be careful of "hitchhiking" data
  - ✓ Pass references to shared files

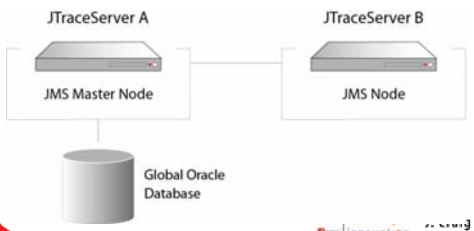
JTraceServer 1.0 Limitations

- JBoss 3.2.3 lacked clustering support for JMS
- All MDBs for one run would execute within a node
- Fork/Join takes advantage of dual CPUs
- If a node goes down
  - ✓ messages queued on that node not processed
  - ✓ client side resend capability allowed processing with reasonable delay.



JTraceServer 2.0 Enhancements

- JBoss 3.2.6 supports single master node clustering for JMS
- All MDBs for one run would execute *across the cluster*
- Fork/Join takes advantage of dual CPUs and multiple nodes
- If a node goes down
  - ✓ messages processed by other nodes in the cluster



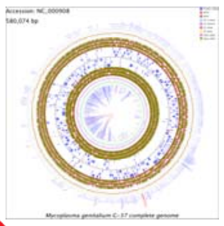
Scalability to the Rescue!

- Challenge
  - ✓ Reprocess 25M traces ASAP
  - ✓ Roughly 6 months of usual throughput
- Solution:
  - ✓ Scale JTraceServer to 8 node cluster (~ 10min)
  - ✓ Insert runs into ProcessingMDB
- Results
  - ✓ Near linear scaling
  - ✓ Job completed painlessly in 2 weeks!!

Microbial Sequencing Website

- Secure access to microbial data
- Scalable bioinformatics application

Microbial Genome http://research.venterinstitute.org



J. Craig Venter
Gordon and Betty Moore Foundation Marine Microbial Genome Sequencing Project
MOORE

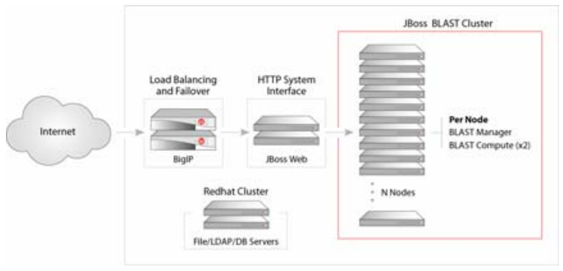
Sequence Comparison Divide and Conquer

- BLAST – Bioinformatics workhorse
- Compare
  - ✓ M subject sequences
  - ✓ N query sequences
- Large searches costly
- Embarrassingly parallel
  - ✓ Divide subject sequences into P parts
  - ✓ Distribute computation
  - ✓ Merge results
- Requirement
  - ✓ Internet accessible blast server
  - ✓ Execute blast jobs asynchronously, quickly

Scalable BLAST – Initial Solution

- Reuse Fork/Join architecture
  - ✓ fork a BLAST job across clustered nodes
  - ✓ Join the results and present to user
- Architecture used:
  - ✓ Stripped down JBoss front end hosts JSPs
  - ✓ JMS-mediated fork/join on compute cluster
  - ✓ Front end drives JBoss cluster via JMS messages
- 2 JMS Queues
  - ✓ BlastManager MDBs service initiation of blast jobs (1)
  - ✓ BlastCompute MDBs execute sub jobs (2)

Scalable BLAST — Architecture



Global Ocean Sampling Expedition

camera

Marine Microbial Ecology

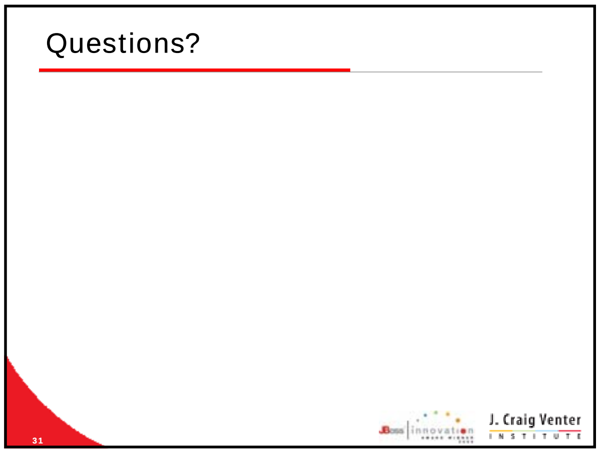
- Driven by growth of metagenomics
- Resource for Marine Microbial Ecology
  - ✓ Funded by Gordon & Betty Moore Fdn
  - ✓ Collaboration with Calit2/UCSD
- Huge compute resource
  - ✓ Managed by Sun Grid Engine (SGE)
- Need massively scalable BLAST
- http://camera.calit2.net

Massively Scalable BLAST

- Scientific Grids Managed by Sun Grid Engine
- Problem:
  Want to scale application by leveraging grid
- Solution:
  - ✓ Adapt fork/join solution
  - ✓ MDBs manage job submission/monitoring to SGE
  - ✓ Fork/join within SGE
- Benefits:
  - ✓ Architectural reuse
  - ✓ Control concurrency on SGE by MDB configuration

Conclusions

- Leverage from using J2EE *everywhere*
- Clustered* JMS a powerful tool for scientific computation
- Fork/Join management with JMS:
  - ✓ Scalability
  - ✓ Robustness
- Massive scaling possible through integration of grid resources



Questions?

31

JBoss innovation award winner

J. Craig Venter INSTITUTE