Enterprise Architecture & Performance

Simon Riggs
Heikki Linnakangas

Overview

- Performance Features in 8.3
- Enterprise Architecture Lifecycle
- OLTP Performance
- Data Integration & Decision Support Performance

Performance Features in PostgreSQL 8.3

- HOT for Frequent Updates*
- Clustered Indexes*
- Database Size Reductions
  - ComboId, Var Varlena
- Checkpoint optimizations
  - mdsync looping avoidance, kill CheckpointStartLock
- XLog File Switch tuning
- Performance Logging
  - log_autovacuum, log_lock_waits
- UnGuaranteed Transactions*
- COPY tuning
- WAL Reductions
  - COPY, CLUSTER
  - B-tree split
- Recovery I/O Reduction
- ORDER BY ... LIMIT
- Merge Join
- L2 Cache Tuning*
  - SeqScan, VACUUM

Enterprise Context

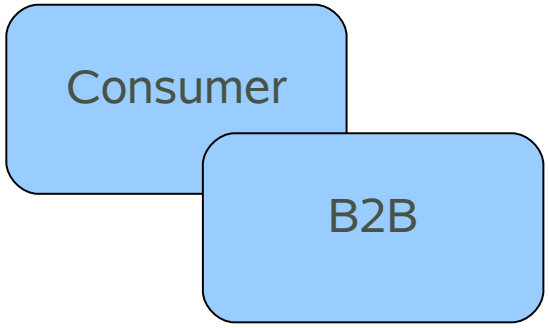
- Which Use Cases do we need to improve?
- Which features do we need?
- Examine some real world situations
- Understand the trajectory of successful PostgreSQL users

Web Site Database

- Single Application
- Single Database
- Early Stage Business Issues
- Time to Market means incomplete functionality

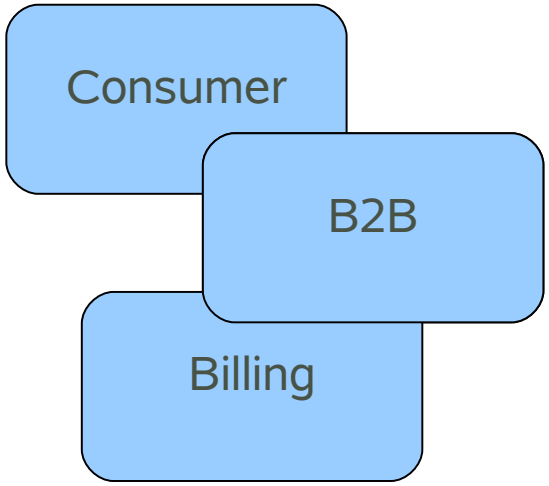
+ Business Sales

- Multiple Customer Roles
- Additional database tables
- New on-line transactions
- Increased complexity
- More variable response times, for some cases



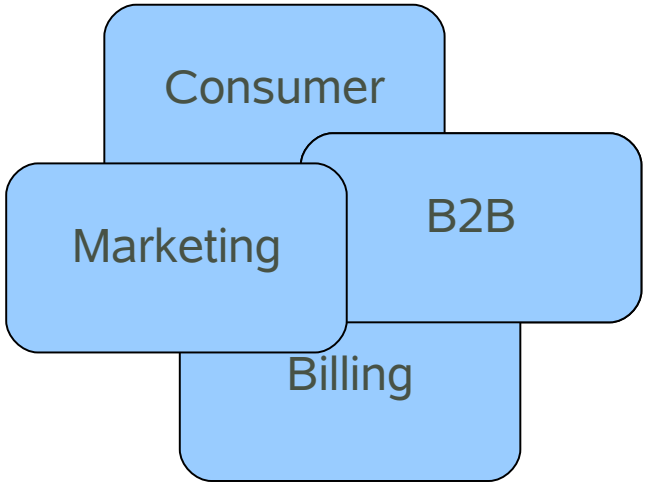
+ Billing

- Business customers want bill consolidation and additional value adds
- Complex bulk tasks
- Very low priority jobs
- Long running transactions



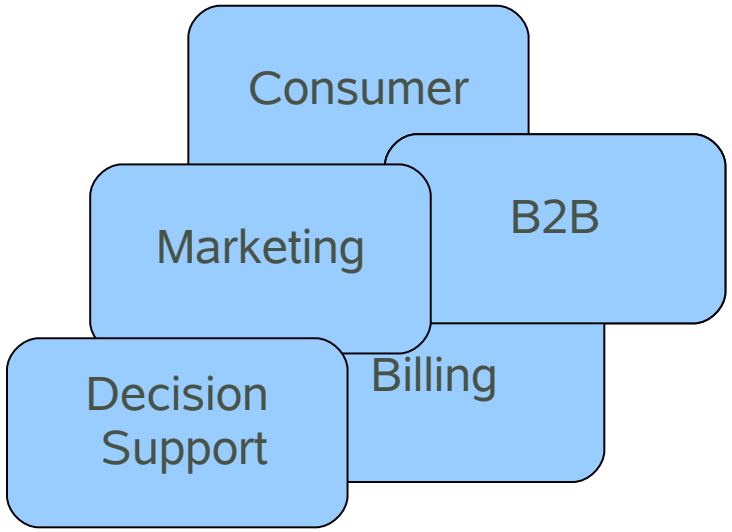
+ Marketing

- Targeted Marketing requires off-line analyses
- Complex, long running analysis tasks, some ad-hoc analysis
- Identification of propensity to purchase/renew
- Pop-up surveys for customer satisfaction measurement
- Adds complexity to other transaction types



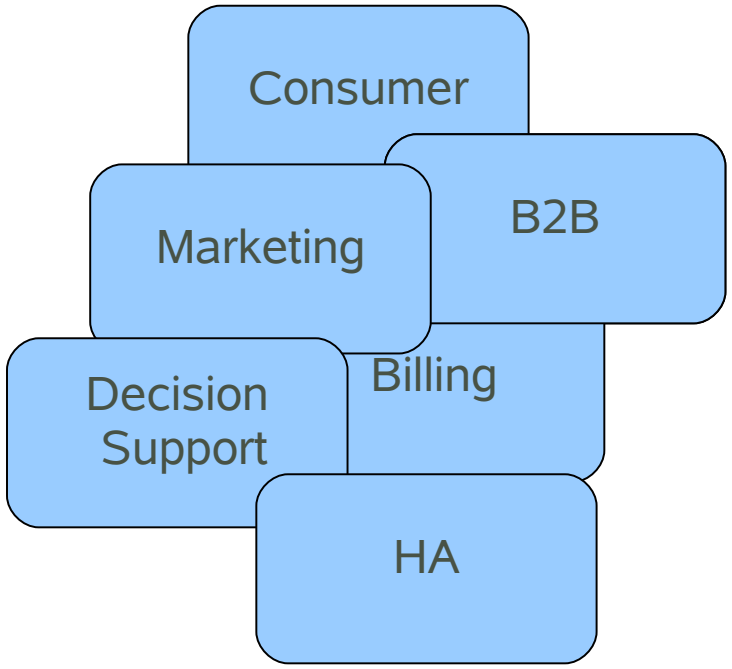
+ Decision Support

- Additional analysis
- Fraud, Churn, Strategic Marketing, Strategic Financial Planning, Due Diligence
- Significant additional workloads for decision support
- Huge additional data volumes
- Long run times



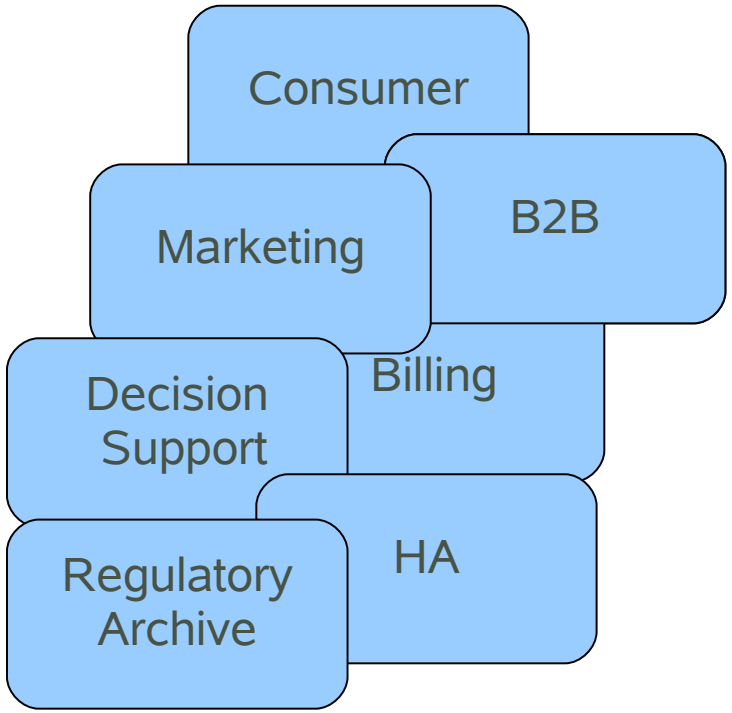
+ High Availability

- Allow for some parts of these databases to be accessible even across hardware outage, preventative maintenance and upgrade



+ Regulatory Archive

- Requirement for massive data storage to meet Government legislation
- Access is rare, yet response time fast when required
- Typically small set of records over a long period of time

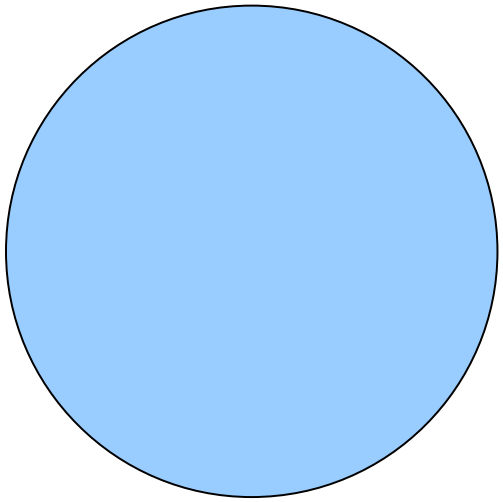


Major Forces on Enterprise Architecture

- Business Growth
- Cost Growth/Reduction
- Workload Evolution
- Mergers & Acquisitions

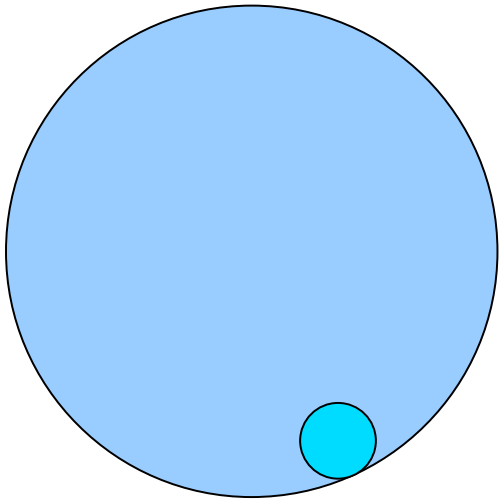
Workload Evolution (1)

- Generic process
- Starting from one reasonably consistent workload



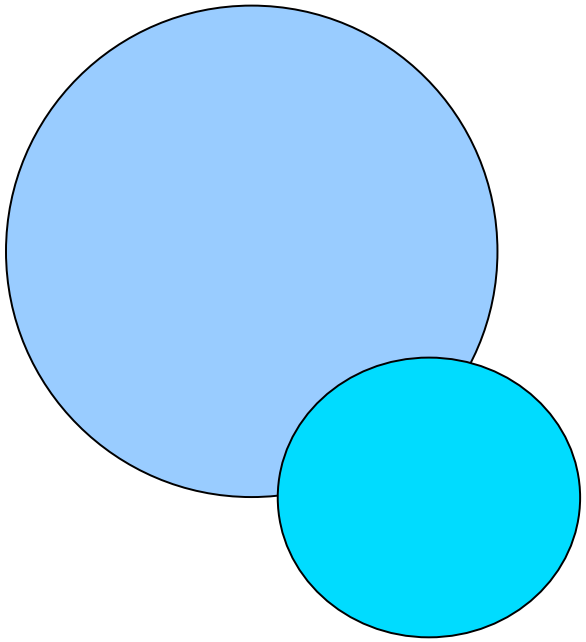
Workload Evolution (2)

- New workload starts
- Shows up as operational and/or performance problems
- Typical Responses
  - “What just happened?”
  - “How do we tune this?”
  - “How do we control this?”
  - “How can we stop this?”



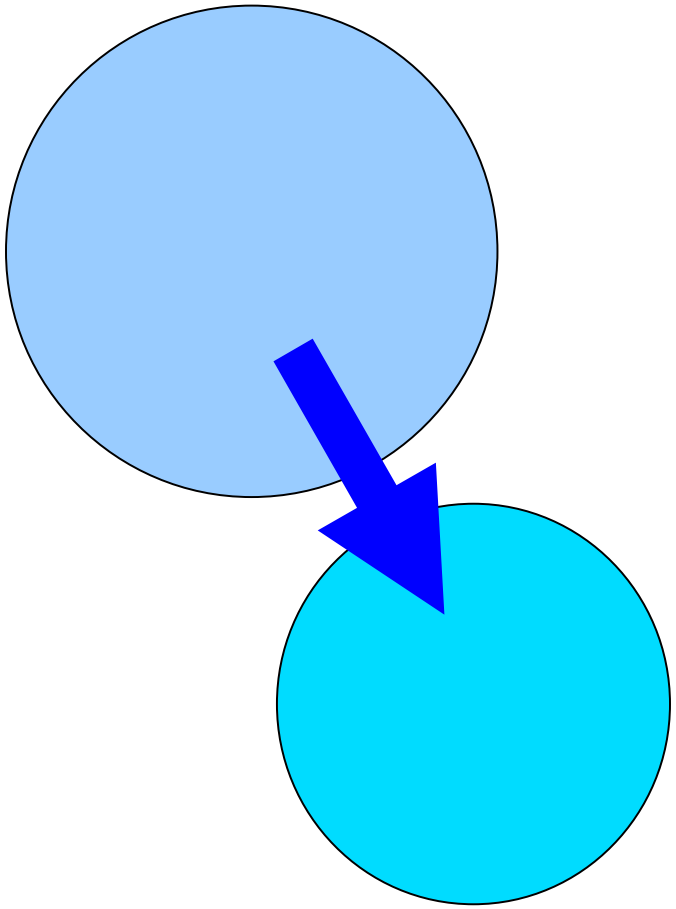
Workload Evolution (3)

- Workload grows
- Recognised as a significantly different workload
- Typical Responses
  - “Can they co-exist?”
  - “How do we prioritise?”
  - “How can we allocate resources effectively?”
  - “When to let them run?”
  - “Can we chargeback?”



Workload Evolution (4)

- Workload separated
- Typical Responses
  - “Best way to transfer data?”
  - “How can we keep them as current as possible?”
  - “How to reduce the impact on the source system?”
  - “How to change the data model as we move the data?”
  - “How can we speed up end-to-end load process?”



Enterprise Architect's Viewpoint

- How can I maintain continuous access to data via web site?
- How can offer a worthwhile Service Level Agreement? How can I do that as the business grows?
- Can I consolidate my servers? How do I mix workloads for as long as possible? How do I approach data integration between servers?
- How can I increase the number of decision support queries the business runs each month?
- How can I mix long term archive with shorter term decision support requirements?

IT Manager's Viewpoint

- We mustn't ever lose our critical data
- The customer must be able to access their data 24x7
- It must be cheap in the long term, not just at the start
- I want it to perform well
- The decision to use PostgreSQL must be low risk. I'm not going to let technical favouritism get in the way of my job.

What is performance, and what can we hackers do about?

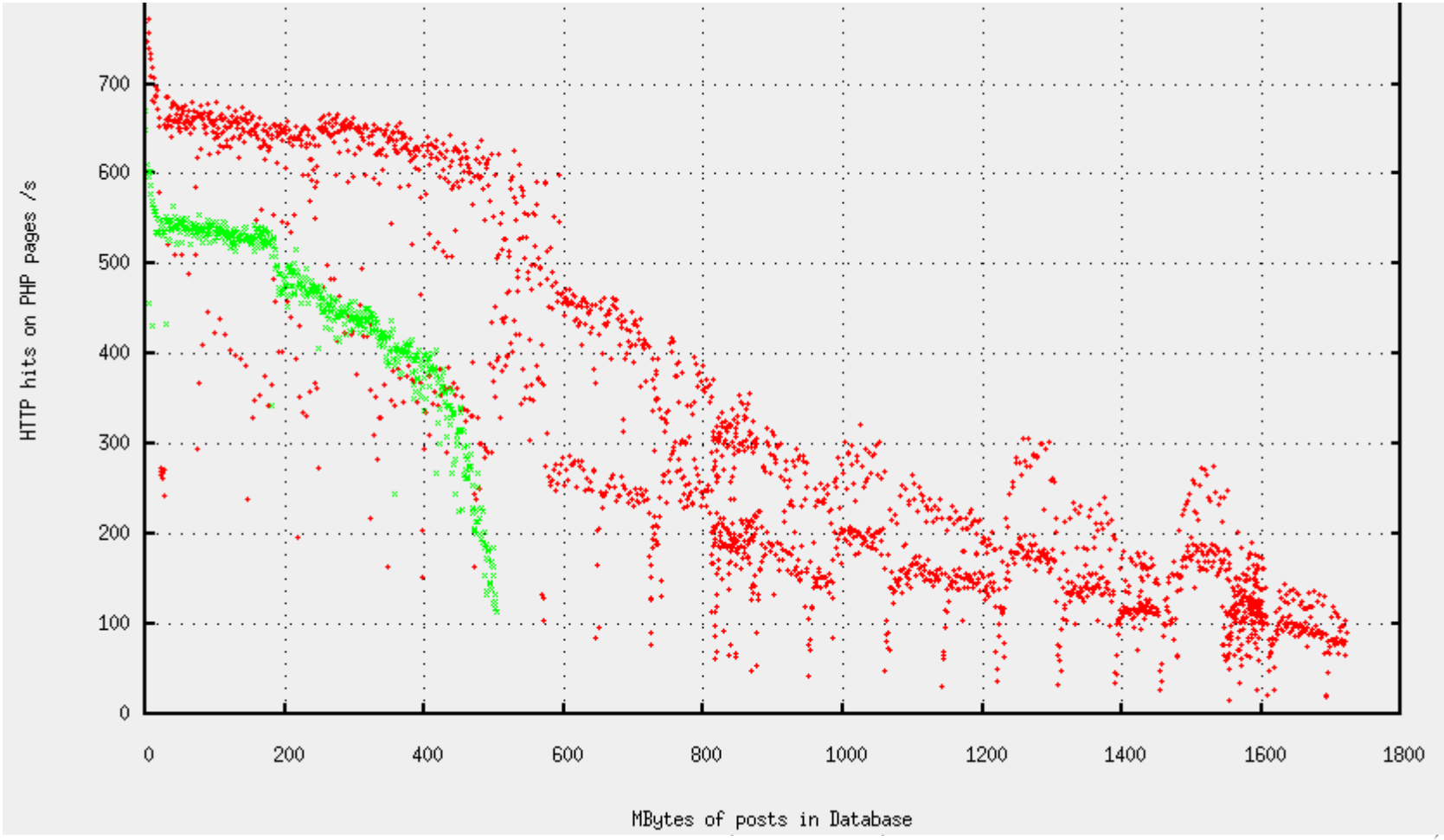
- "I want it to perform well"
- Performance
  - **Steadiness**
  - Throughput
  - Adaptation
- "It must be low-risk" -> Scalability

Steadiness matters

- Unsteady performance is often worse than consistently poor performance
- “Every now and then the system stops responding, but revives after a few minutes. What's going on?”
- Unsteadiness makes benchmarking harder
- Also an availability issue: If the system stops responding, it's not available

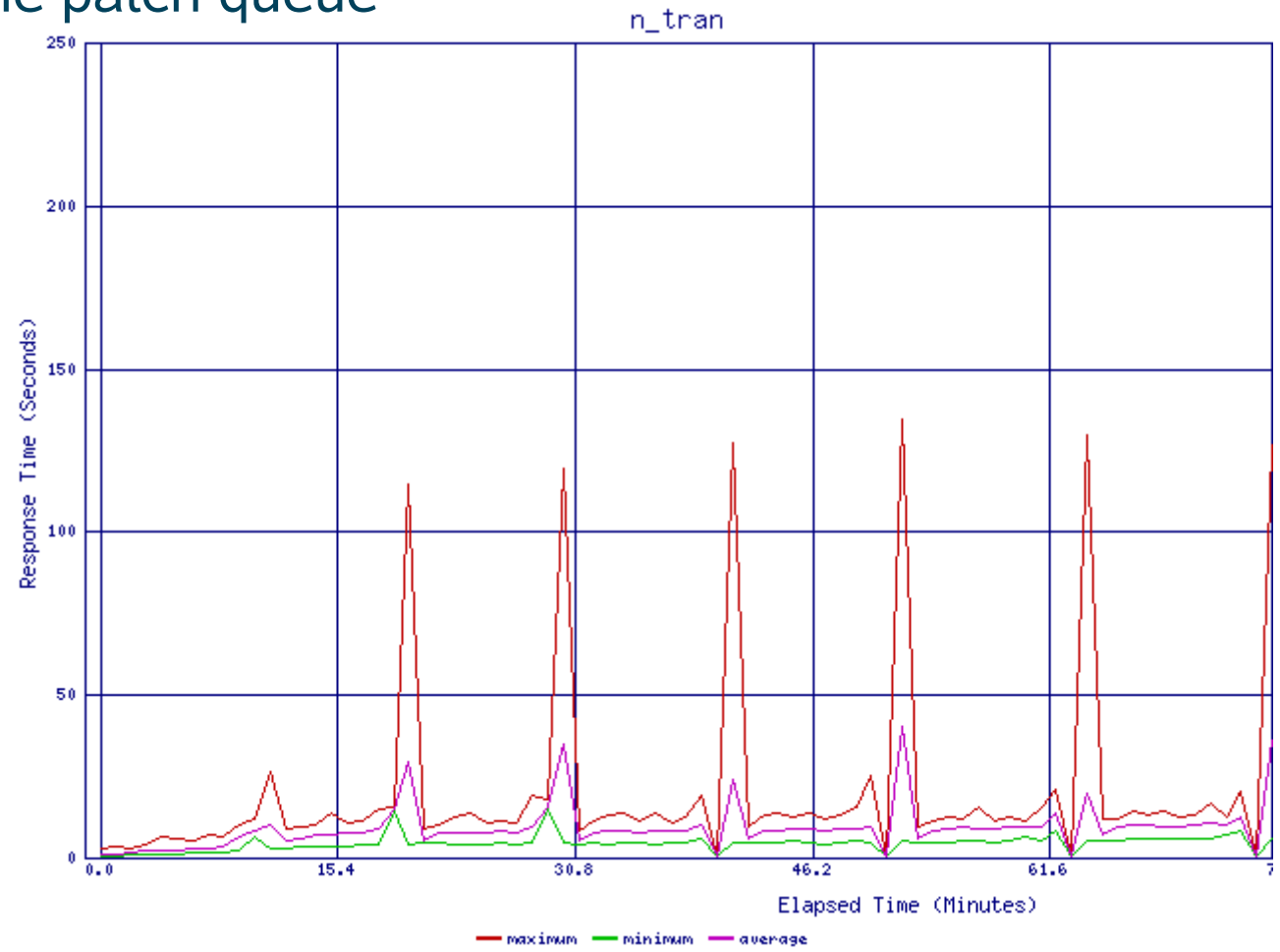
Steadiness matters

- Benchmark of MySQL and PostgreSQL by PFC, per mail titled “Postgres Benchmark Results”, 20 May 2007:



Sources of unsteadiness: Checkpoints

- Load distributed checkpoints patch by Itagaki Takahiro is in the patch queue



Sources of unsteadiness: Vacuum

- Vacuum has a significant impact on concurrent activity
  - vacuum_cost_delay helps, but needs manual tuning
- While autovacuum is vacuuming a large table, smaller ones are neglected
- Vacuum doesn't scale nicely

Future of Vacuum

- Make vacuum cheaper
  - Dead space map
  - Skip 2nd vacuum pass, set xvac in the 1st phase instead
  - Piggyback vacuum on normal database activity
- Reduce the needed vacuum frequency
  - HOT
  - Truncate dead tuples to line pointers
- Retail vacuum
  - Reference counting

Sources of unsteadiness: Other queries

- Concurrent queries have an impact on other queries
- In patch queue
  - Make buffer cache scan-resistant
  - synchronized seq scans by Jeff Davis

Sources of unsteadiness: Plan Instability

- Cached plan becomes ineffective when data changes
- A sub-optimal plan is chosen when data changes

- Future research:
  - Invalidate plans when statistics are updated
  - Improve cost estimates
  - Assigning a confidence factor to estimates
  - Estimate hints, like marking tables as volatile

Throughput

- Performance
  - Steadiness
  - **Throughput**
  - Adaptation

Adaptation

- Performance
  - Steadiness
  - Throughput
  - **Adaptation**
- Availability

Adaptation

- “Can we run OLTP and DSS on a single server?”
- “I don't want to hire a consultant to tune the system for a week”

- “*Out-of-the-box*” performance
- System needs to adapt to changing workload, without having to manually change GUC parameters and restarting.

- The ultimate goal is to get rid of all performance-related GUC parameters

Adaptation: Memory management

- shared_buffers
- work_mem
- wal_buffers
- temp_buffers
- max_fsm_pages & max_fsm_relations

Adaptation: Background Writer

- Autotuning bgwriter options for 8.3:
  - Greg Smith and Itagaki Takahiro

Adaptation: Checkpoints

- Requirements for checkpoint_interval and checkpoint_segments really comes from a more fundamental business requirement:
  - Maximum acceptable recovery time

- Checkpoint intervals should be expressed in terms of that.

Availability

- Performance
  - Steadiness
  - Throughput
  - Adaptation
- **Availability**

Availability

- “The customer must be able to access their data 24x7”
- No maintenance window
- Implies a reasonable response time -> steadiness

Availability: Recovery time

- Faster recovery with full_page_writes=on in 8.3
- Future research:
  - Reducing WAL size
  - COPY performance (to speed up pg_restore)

Availability

- Changing parameters
  - Many parameters need a restart for change to take effect.
- Offline maintenance tasks
  - CLUSTER
  - VACUUM FULL
    - CLUSTER is MVCC-safe in 8.3, making it a more viable alternative to VACUUM FULL.

Data Integration & DSS Performance

- Log-based Replication
  - Reducing cost of data integration
  - Data Integration and High Availability
- Read-only tables
  - Long term archiving
  - Optimised partitioning
  - Increased Referential Integrity performance

Summary

- Holistic feature planning can help PostgreSQL Hackers
- Success trajectories in major Enterprises are a good model
- Performance is a many headed beast