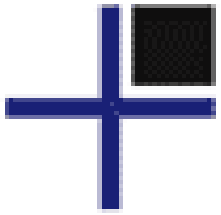
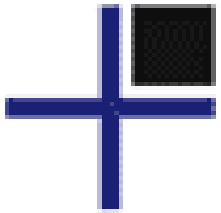


PGCon 2011

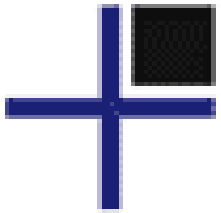
PostgreSQL Performance Pitfalls

Greg Smith - © 2ndQuadrant US 2011



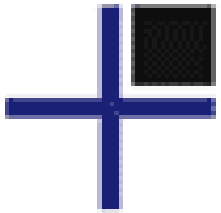
Too much information

- PostgreSQL has a FAQ, manual, other books, a wiki, and mailing list archives
- RTFM?
- The 9.0 manual is 2435 pages
- You didn't do that



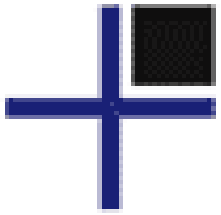
PostgreSQL Version Policy

- 8.3 → 8.4: Major version upgrade
- 8.4.3 → 8.4.4: Minor version upgrade
  - No feature changes
  - Bug fixes only
  - Can involve database corruption
  - Backports: more risky to *not* have the change
  - Other vendors might say:
    - “Fix pack”
    - “Service pack”
    - “Hot fix”
  - Stay as current as possible



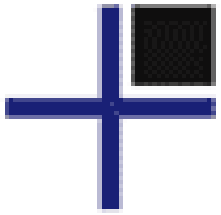
Starting Version

- Major changes in PostgreSQL 8.3
- Upgrades from earlier ones very painful
  - In-place upgrades become possible
- Performance is much better in 8.3
- New projects shouldn't consider anything earlier
- Your operating system packages are *not* better
  - Same packagers involved in many cases



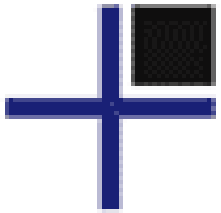
Default configuration

- Optimized for startup with low shared memory
- Many parameters too low for your phone!
- Need to adjust several memory related items
- http://wiki.postgresql.org/wiki/Tuning_Your_PostgreSQL_Server
- pgtune



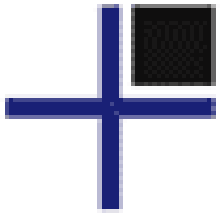
Major parameters to set

- shared_buffers: 512MB to 8GB
- checkpoint_segments: 16 to 256
- effective_cache_size: typically ¾ of RAM
- work_mem: memory / connections / (4 to 32)



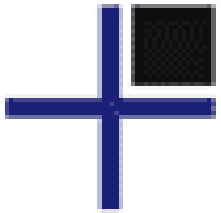
Bad table statistics

- “Why didn't it use my index?”
  - Sequential scans can be faster
- Might be statistics issues however
- Check data from EXPLAIN ANALYZE
- Look for variation between estimated vs. actual
- Manual ANALYZE doesn't take very long
- May also have to increase statistics target
  - Can be done on a single column



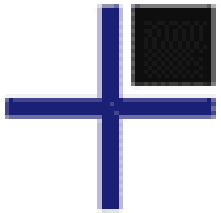
Statistics adjustment

- default_statistics_target is 100 in >=8.4
- 100 – 1000 is useful range
- ALTER TABLE t COLUMN c SET STATISTICS n;
- Check analyze dates in pg_stat_user_tables
- Study pg_stats, learn about MCVs and histograms



VACUUM

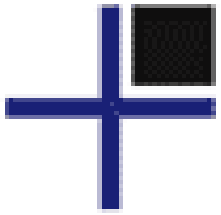
- Cleans up after UPDATE and DELETE
- Autovacuum updates database statistics
  - Also considers INSERT quantity
- Large tables: 20% change required
- Intensive when it happens
- Must happen eventually for frozen ID cleanup
- Focus on smoothing and scheduling, not delay
- Dead rows add overhead you just don't see
  - Table “bloat” can be very large



VACUUM monitoring

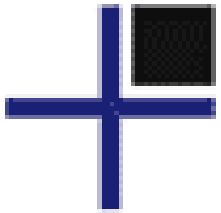
- Watch timestamps in pg_stat_user_tables
- Beware long-running transactions
- log_autovacuum_min_duration
- Sizes of tables/indexes critical too

http://wiki.postgresql.org/wiki/Disk_Usage



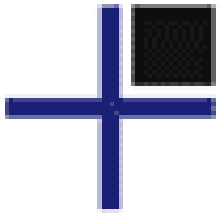
Index Bloating

- Indexes can become less efficient after deletes
- VACUUM FULL before 9.0 makes this worse
- REINDEX helps, but locks too
- CREATE INDEX can run CONCURRENTLY
  - Rename to simulate REINDEX CONCURRENTLY
- CLUSTER does a full table rebuild
  - Same “fresh” performance as after dump/reload
  - Full table lock to do it



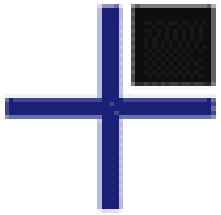
Extensions and contrib

- Ship with the server
- Not installed by default
- Extensions are easy to install starting in 9.1
  - May need postgresql-contrib package
- Many useful ones in earlier versions too
  - Just hard to install
- Page/index inspection tools



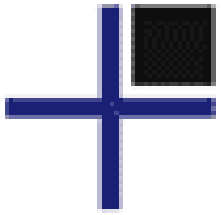
Useful extensions

- pageinspect pgstattuple pg_freespacemap
- pgrowlocks
- pg_stat_statements (8.4)
- auto_explain (8.4)
- pg_archivecleanup (9.0)
- pgcrypto
- dblink
- hstore



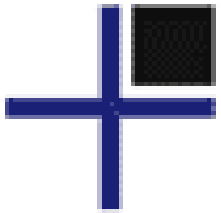
External tools

- You need them
- The PostgreSQL “core” is just that
- Many essential add-on tools



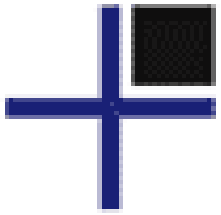
Monitoring

- Can use OS tools for simple monitoring
- You also want to monitor database stats
  - pg_stat_user_tables most essential
  - Helpful to track table/index sizes too
- Best tools combine OS data with database
  - Munin, Cacti



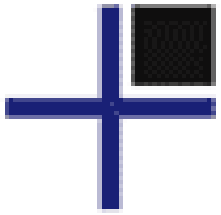
OS Monitoring

- top -c
- vmstat 1
- iostat -mx 5
- htop
- iotop
- watch



Incident capture

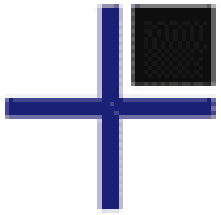
- top -c -b
- nohup
- script



Table/Index Sizes

```
SELECT nspname,relname,
    pg_size_pretty(pg_relation_size(C.oid))
      AS "size",
    pg_size_pretty(pg_total_relation_size(C.oid))
      AS "total_size"
  FROM pg_class C
  LEFT JOIN pg_namespace N
    ON (N.oid = C.relnamespace)
  WHERE nspname NOT IN
    ('pg_catalog', 'information_schema')
  ORDER BY pg_relation_size(C.oid) DESC;
```

More at http://wiki.postgresql.org/wiki/Disk_Usage

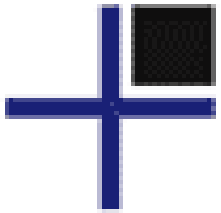


Locking

- Need to combine pg_locks + pg_stat_activity
- Example queries:

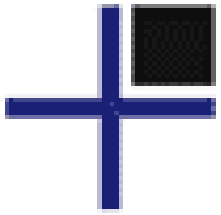
http://wiki.postgresql.org/wiki/Lock_Monitoring

- Surprising lock acquisition type and order



Database Monitoring

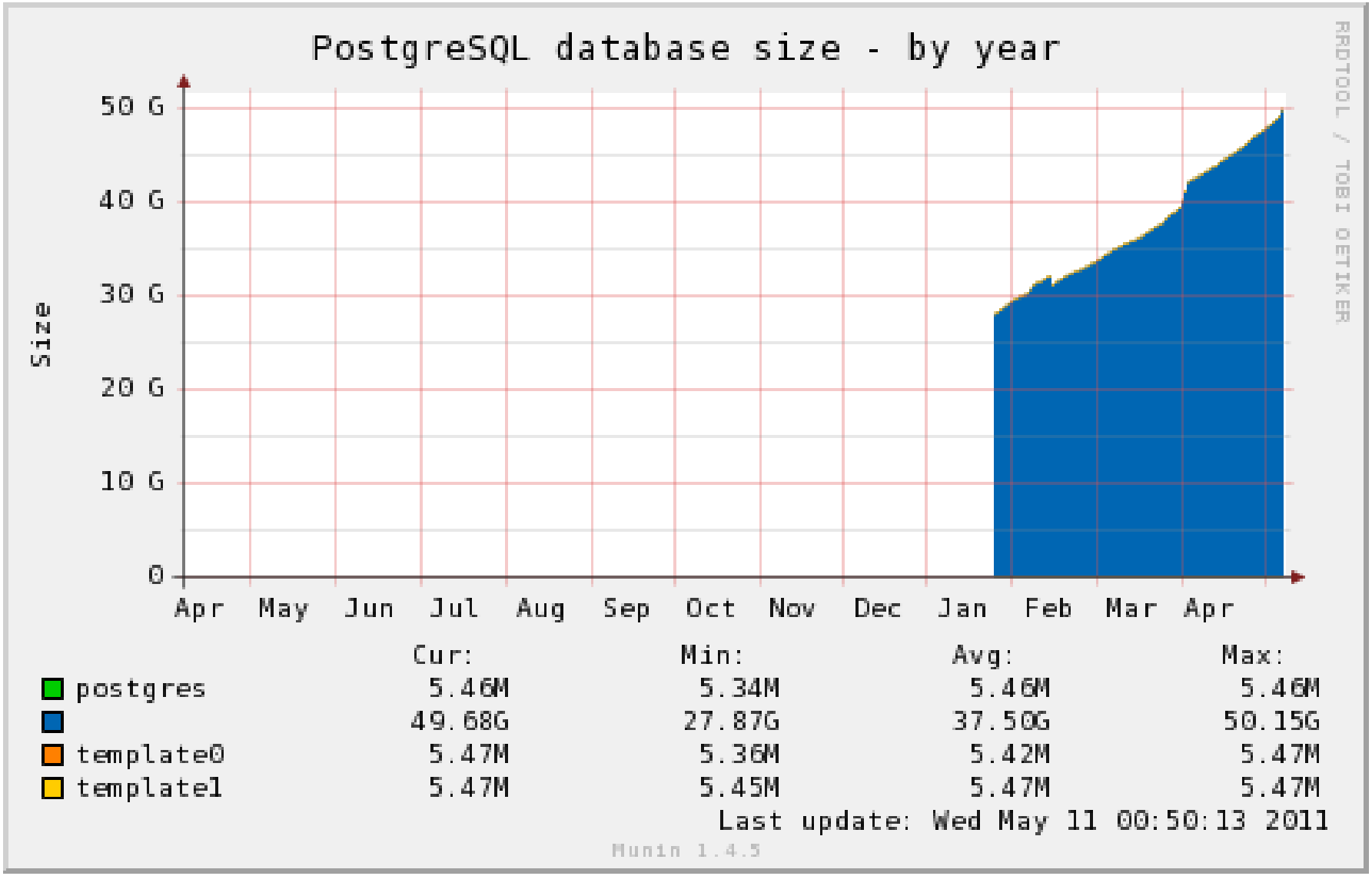
- top -c
- pg_stat_activity
- pg_locks
- pg_top

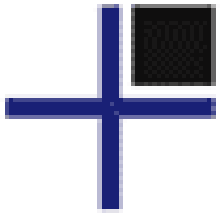


Alerting

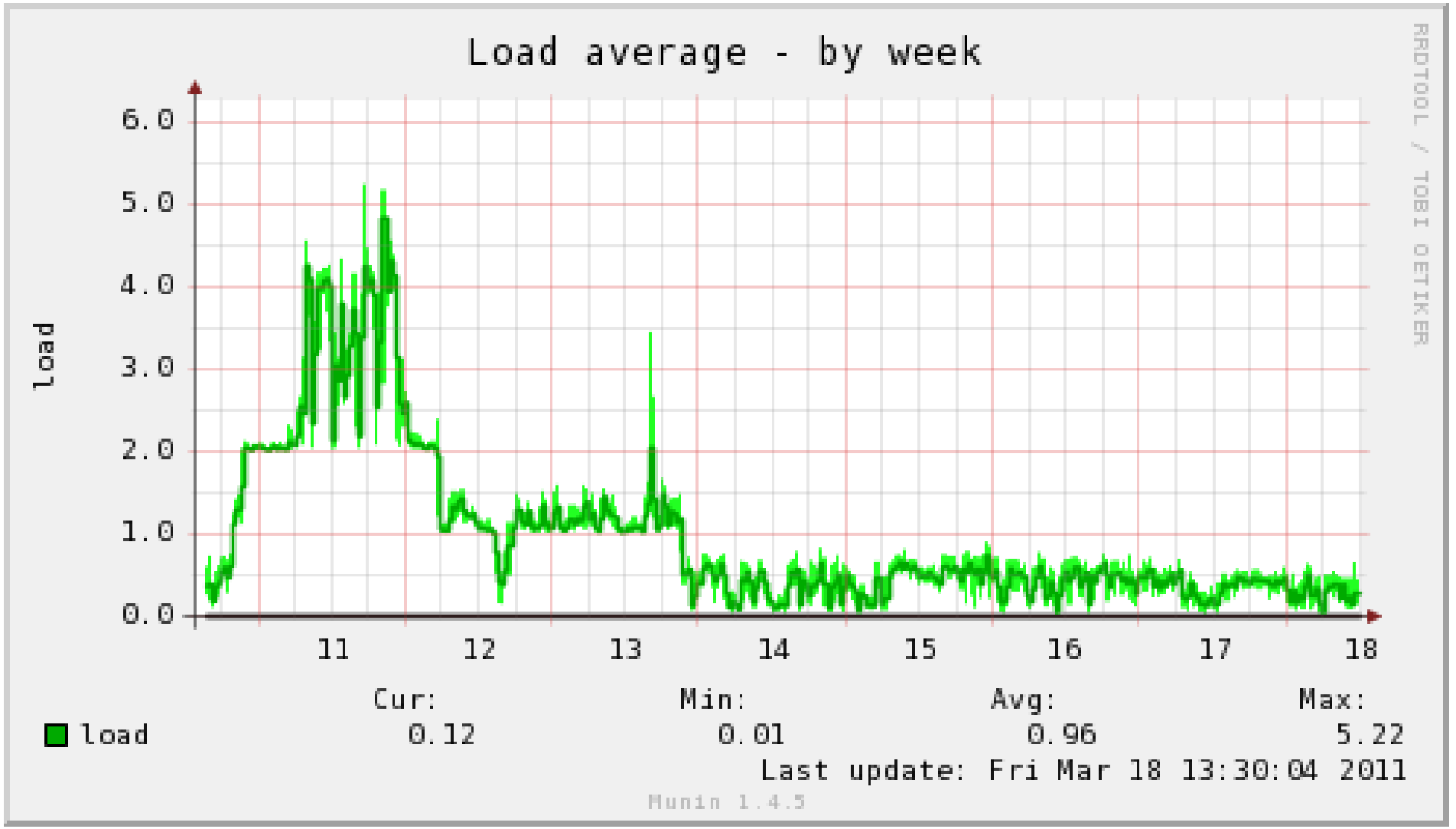
- Out of disk space? Server will crash.
- pg_xlog WAL data should be small
  - Often very small filesystem, easy to fill
- Critical on standby servers too
  - Watch size of backup archive
- check_postgres for Nagios, others

Trends matter

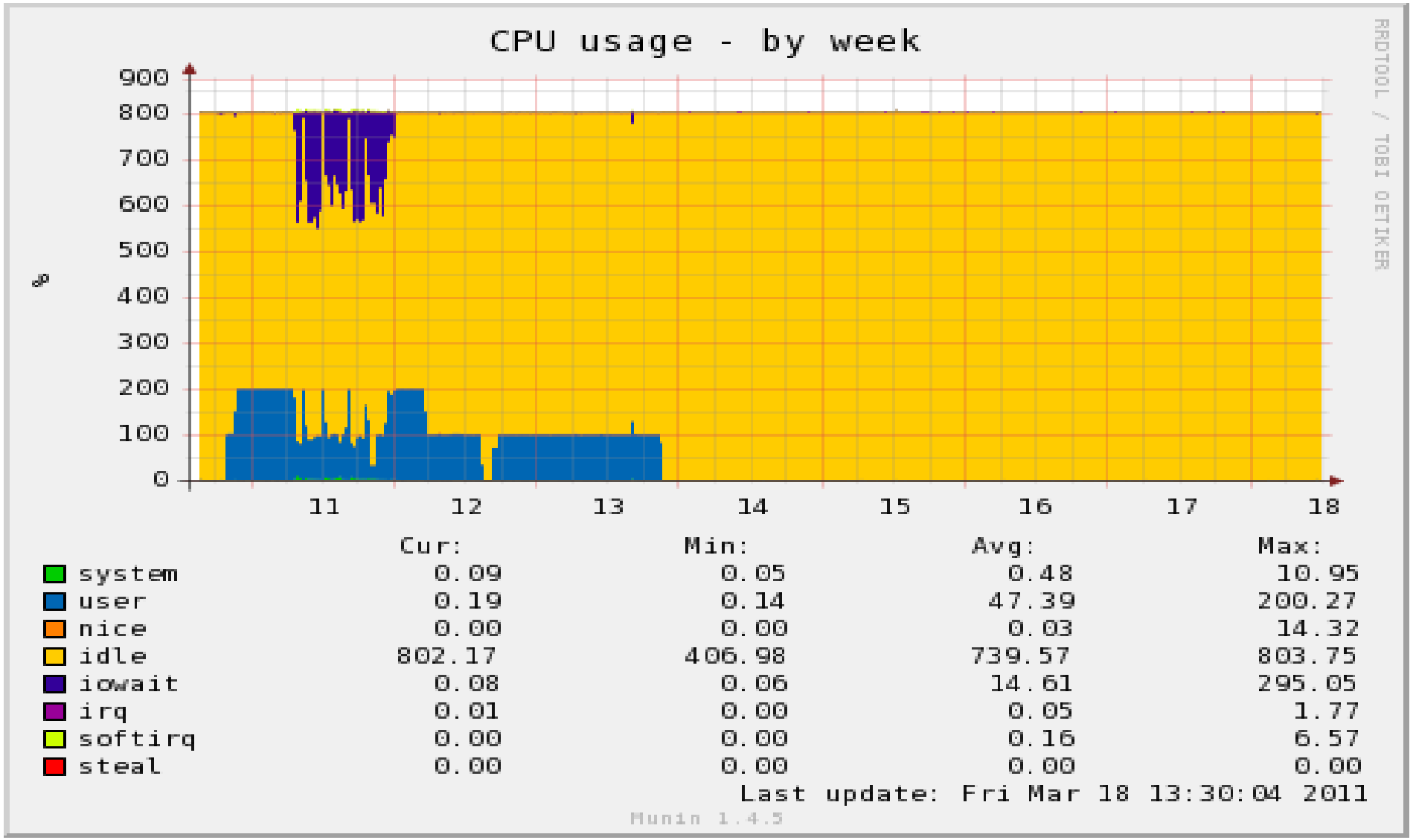




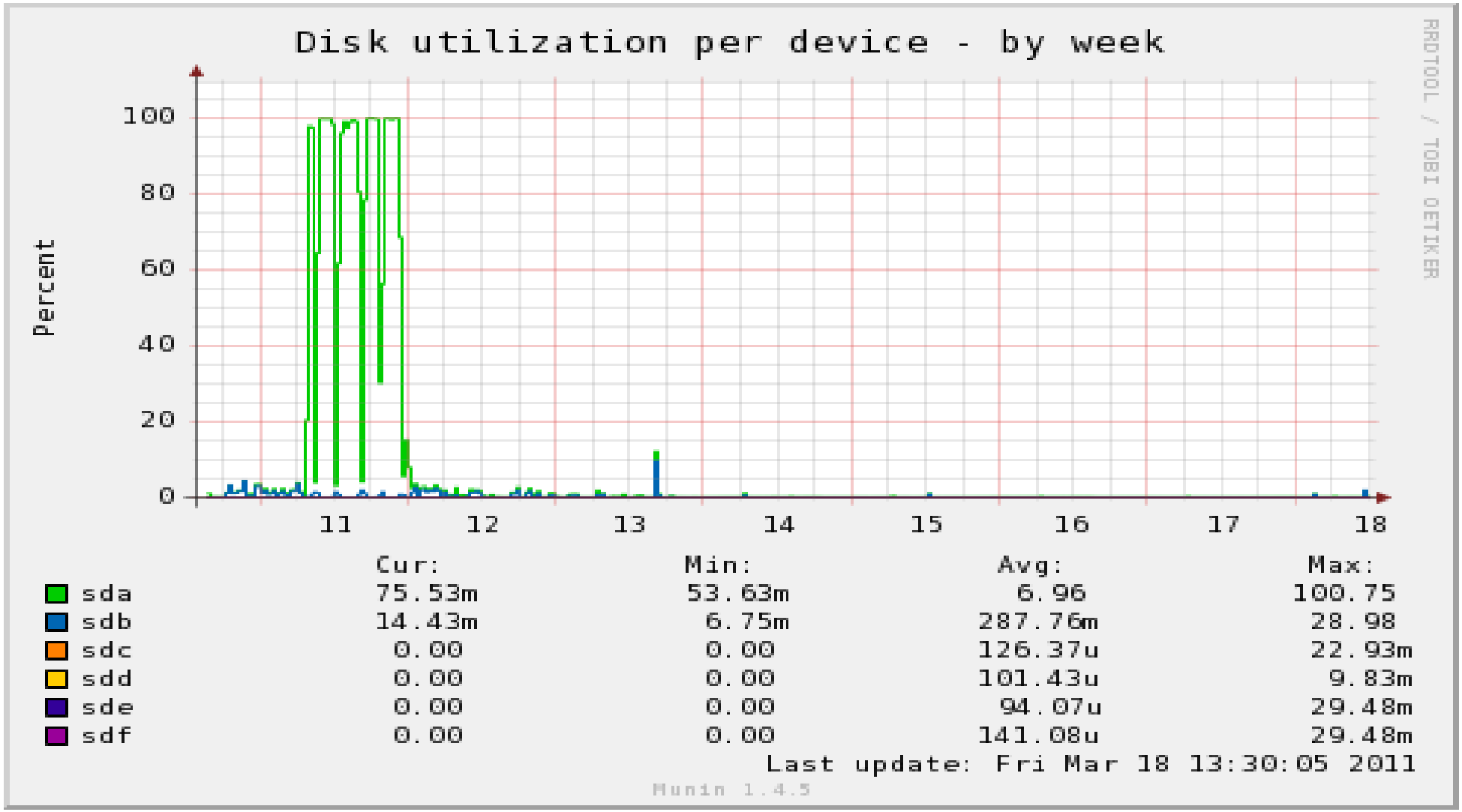
Munin: Load average

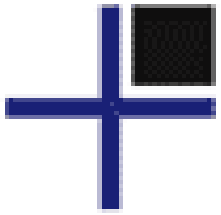


Munin: CPU usage

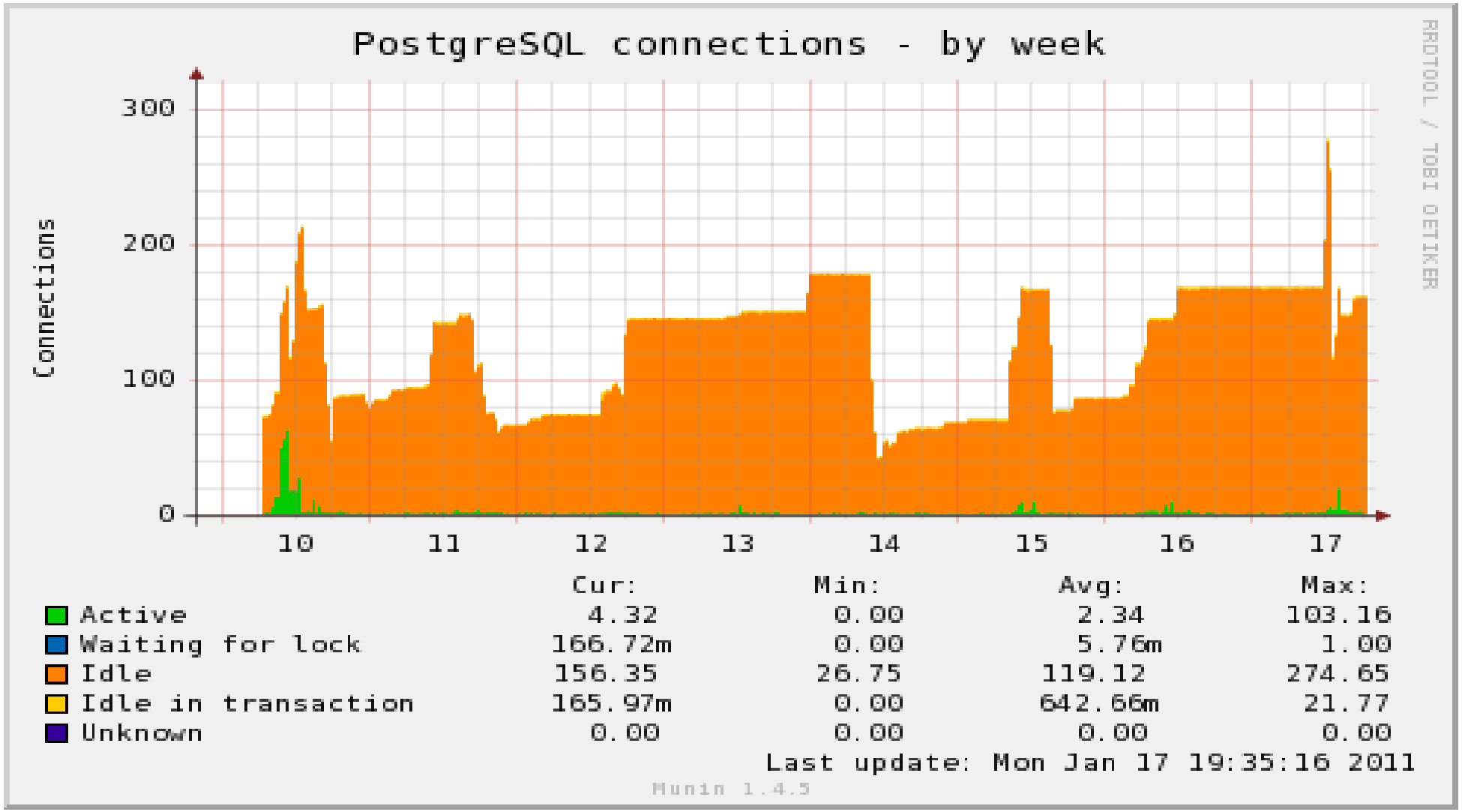


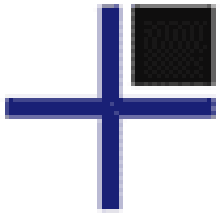
Munin: Disk Util %



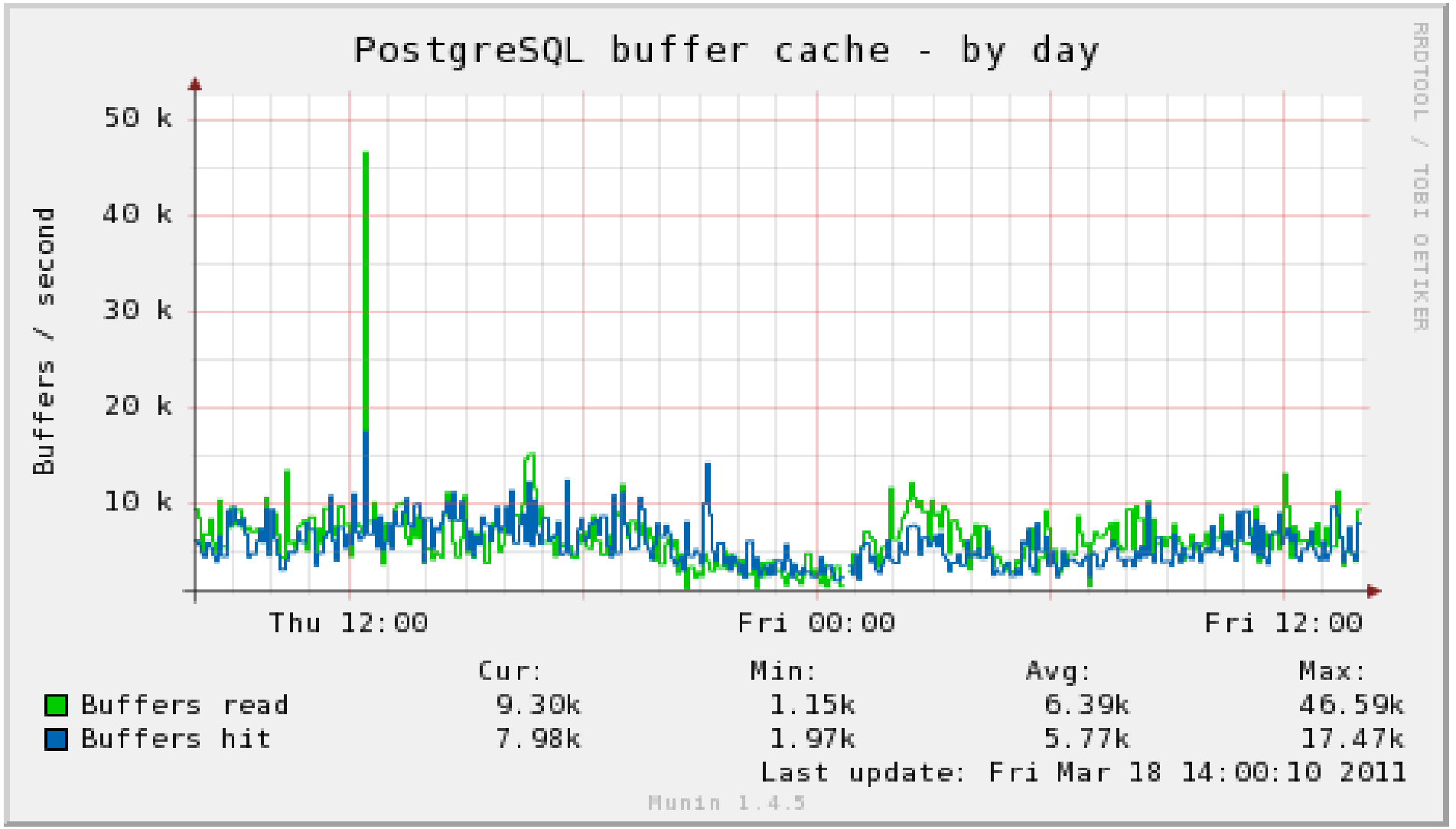


Connection Distribution

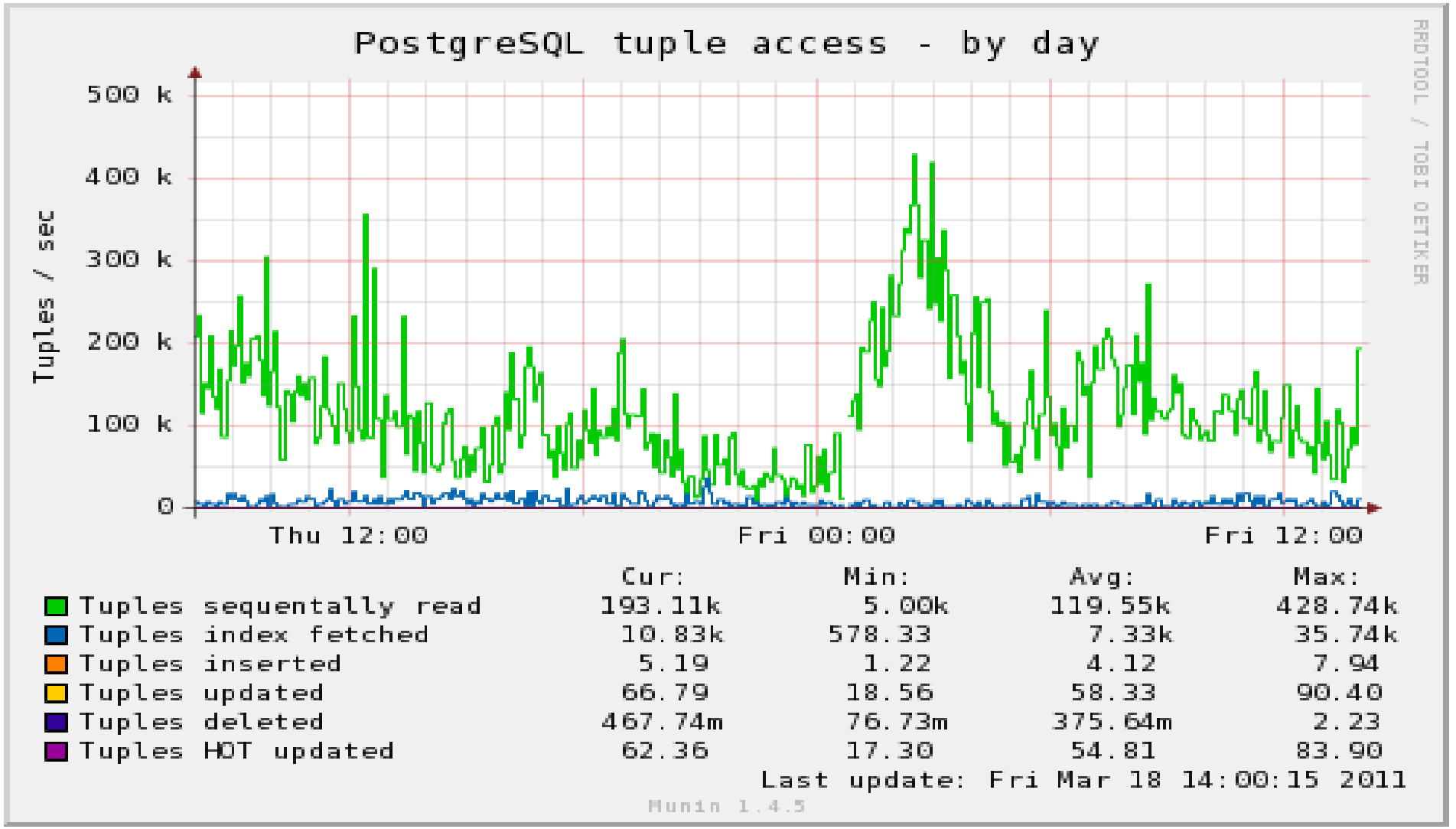




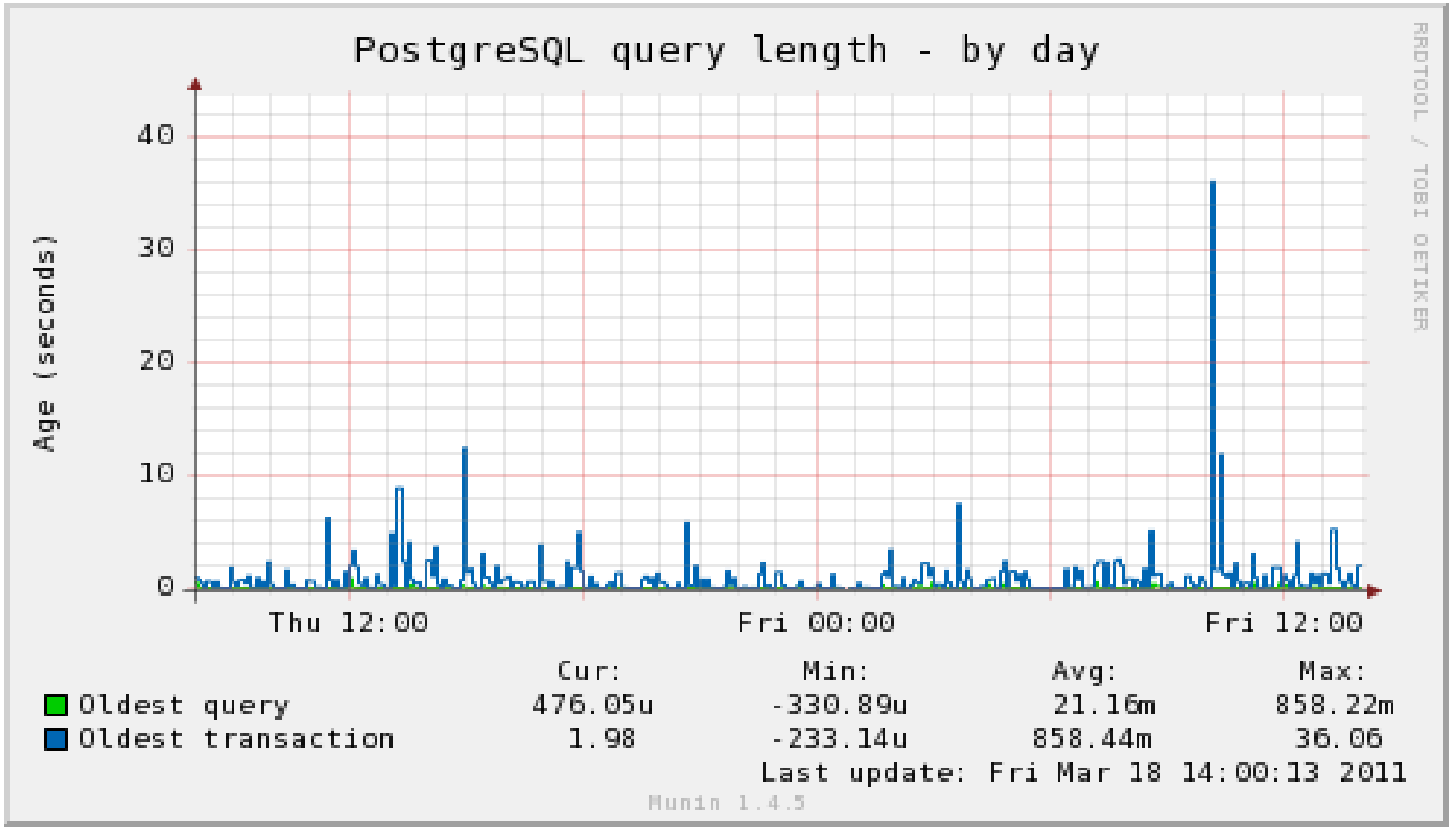
Database shared_buffers

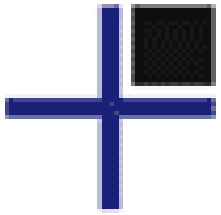


Workload distribution



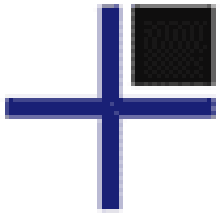
Long queries/transactions





Query Log Analysis

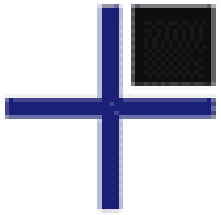
- Collect log data
  - log_min_duration_statement
  - Auto-explain
- Built-in pg_stat_statements
- Analyze logs with external tool
  - pgFouine



pgFouine

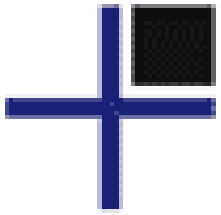
Queries that took up the most time (N) ^

Rank	Total duration	Times executed	Av. duration (s)	Query
1	**1933h26m41s**	23,387	297.62	**UPDATE** accounts **SET** filler=*lower*('') **WHERE** aid < 0; Show examples
2	**17h14m20s**	23,387	2.65	**UPDATE** branches **SET** filler=*upper*(''); Show examples
3	**17m13s**	23,387	0.04	**SELECT** history.* **FROM** accounts, history **WHERE** accounts.aid=0 **AND** accounts.aid=history.aid; Show examples
4	**15m4s**	23,387	0.04	**SELECT** accounts.* **FROM** accounts,history **WHERE** history.aid=0 **AND** accounts.aid=history.aid; Show examples



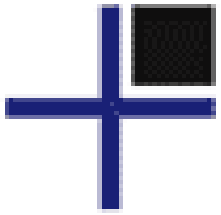
Defensive, preemptive logging

```
log_line_prefix = '%t [%p]: [%l-1] user=
  %u,db=%d '
log_min_duration_statement = 1000
log_temp_files = 0
log_checkpoints = on
log_connections = on
log_lock_waits = on
log_autovacuum_min_duration = 1000
```



Connection Pooling

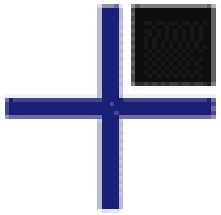
- Connections are expensive
- >500 at once will never work well
  - Windows dies at ~125
- Swapping between many processes is slow
- Optimal connections ~2-3X core count
- Use a connection pooler to limit connections
  - pgbouncer, pgpool-II, application server



Resource limits

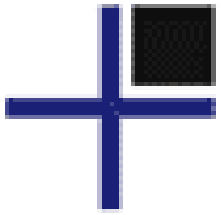
- Memory limits aren't just a work_mem issue
- Tough to balance across the server
- No way to limit one query in the short term
- statement_timeout
- Use nice, ionice; more at

http://wiki.postgresql.org/wiki/Priorities



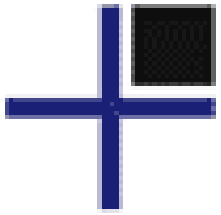
Replication surprises

- Built-in replication is not statement based
- Whole cluster only, not per-database
- Read-only queries can be send to standby
- But they can be canceled
- Better features here in 9.1



Low level issues

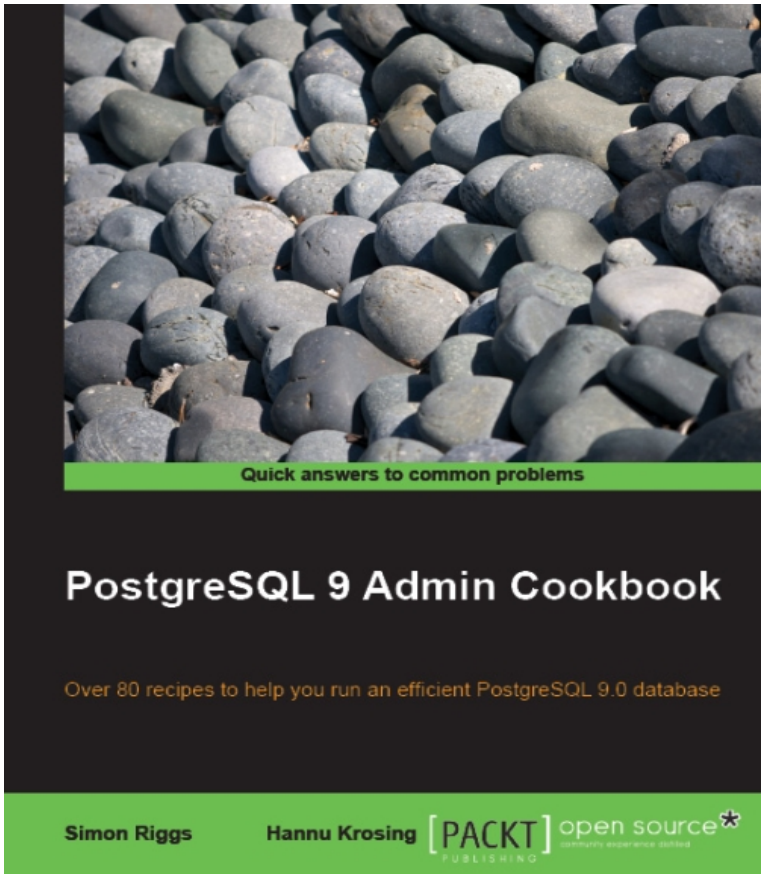
- Log checkpoints to catch spikes
- Watch lock waits
- Constraint and foreign key overhead
- Overindexing
- Linux Out-of-memory killer
- Battery-backed write cache for fast commits
  - Beware volatile write caches

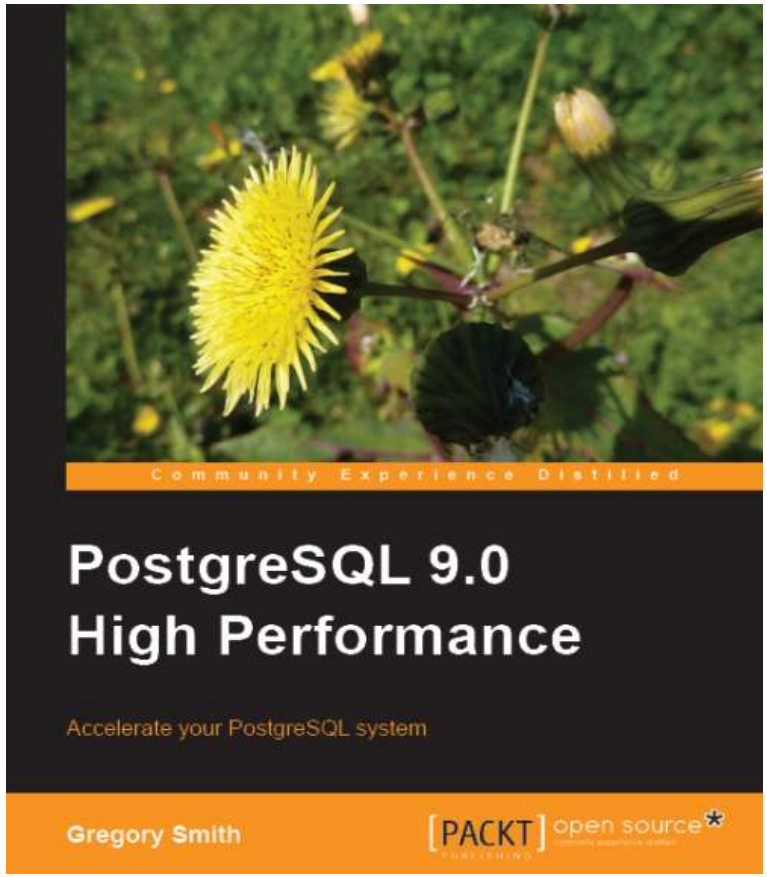


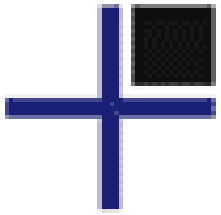
Questions?

PostgreSQL Books

http://www.2ndQuadrant.com/books/







2ndQuadrant Training

- Available now in UK, Italy, Germany
- Coming this fall to the US
- Includes hands-on workshops with instructor interaction
- Database administration
- Performance
- Replication and Recovery
- Real-world experience from production DBAs