Building and Deploying Web-Scale Social Networking Applications, Using PHP and Oracle Database

Oracle Open World San Francisco

September 23, 2008

Who is Community Connect?

Community Connect Inc. (CCI) is the social networking division of Interactive One, a Radio One Company.

- Founded in 1996 w/ the launch of AsianAvenue.com

- Launched five niche social networking sites, the largest of which is BlackPlanet.com (aka BP)

- Combined traffic of over 600 million page views/ month across all sites, user base of over 25 million members

- Acquired by Radio One in April 2008

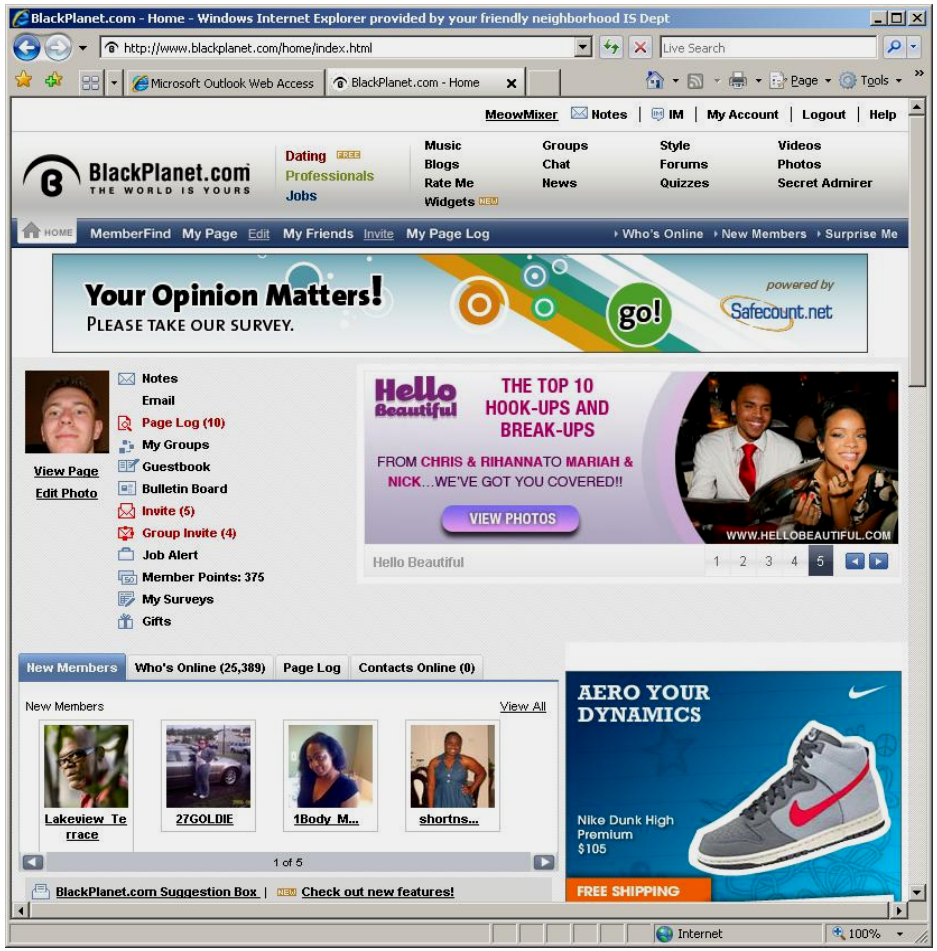
Who are we?

Levi Dixon
Senior Database Architect
(Former Developer, Former SA)
Joined CCI in 07/2000
ldixon@communityconnect.com

Nicholas Tang
VP, Technical Operations
Joined CCI in 03/2001
ntang@communityconnect.com

BlackPlanet.com (BP)

BlackPlanet.com is CCI's largest online property, with over 19 million users and over 500 million page views per month.



(BP) Web Traffic

BlackPlanet.com receives over 500 million page views in an average month (over 1 billion total requests per month)

(BP) Workload

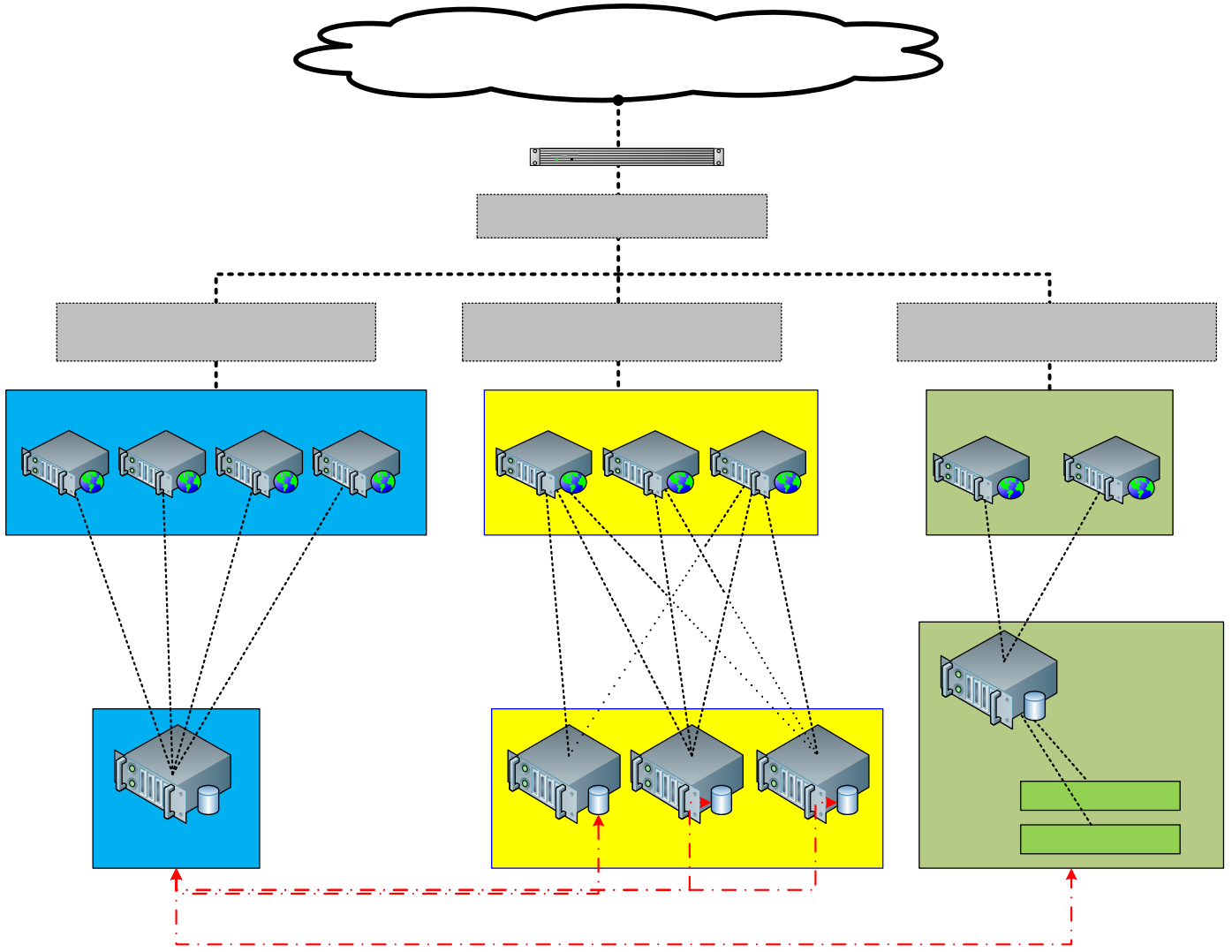
BlackPlanet.com's infrastructure includes over 100 web servers and 15 databases.

- Red Hat Enterprise Linux 4.3, Oracle Enterprise Linux 5.2
- Apache 2
- PHP 5.2.4, OCI8 extension 1.3.2 beta, APC 3
- Oracle 10*g*R2 – 11*g*R1; 10*g* RAC; 11*g* RAC

Peak traffic:

- 1,700 dynamic web requests/ second
  - HTTP pages; AJAX requests not included
- 2,000 transactions / second
  - 13,000 executions / second
    - Primarily OLTP
    - More reads than writes
- 10-15 database instances / site

(BP) Basic Infrastructure



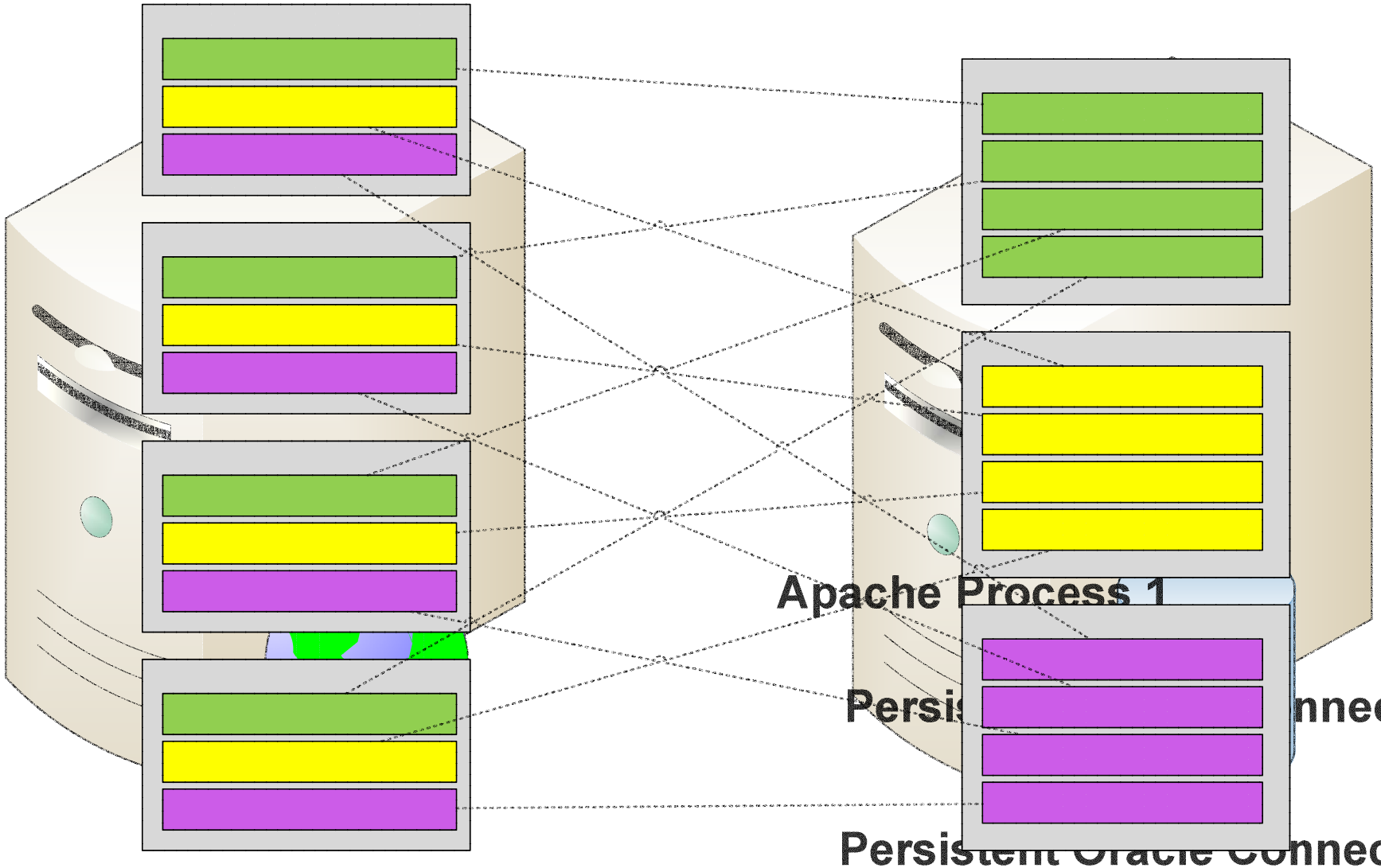
(BP) The Distributed Database Problem

- Apache/ PHP maintains open connections to each database schema
  - 50,000 users online = 50,000 HTTP keepalive connections
  - Hardware load balancer does TCP connection pooling
  - Results in average of 5000 Apache processes for BP
  - 25 DB schemas (1-4 schemas per instance)
  - Up to 125,000 persistent connections across the BP databases
    - 1 persistent connection to Oracle = 1 shadow process = ~5MB RAM on Oracle database (depending on RDBMS parameters)
  - 625 GB RAM required across cluster to support max shadow processes
  - $500/GB of RAM (2002) = over $300,000 worth of extra RAM (on top of base RAM for OS and Database)

Why so many schemas?

- Lots of different types of applications for end users on the site
- Schemas are split within a site by “application function” in most cases. E.g.
  - Groups
  - Photos
  - Forums
- Allowed for unanticipated growth for new applications
  - E.g. The next 4 new apps can go on one instance, and be split later based on usage patterns
  - Allowed schemas/applications that grew quickly to be “relocated” to a different instance if they outgrew their database
  - Allowed web servers to be clustered to only serve pages that access one schema
- Exacerbated the shadow process problem (in general)
  - We have since severely limited addition of new schemas and consolidated several schemas to try to find a balance

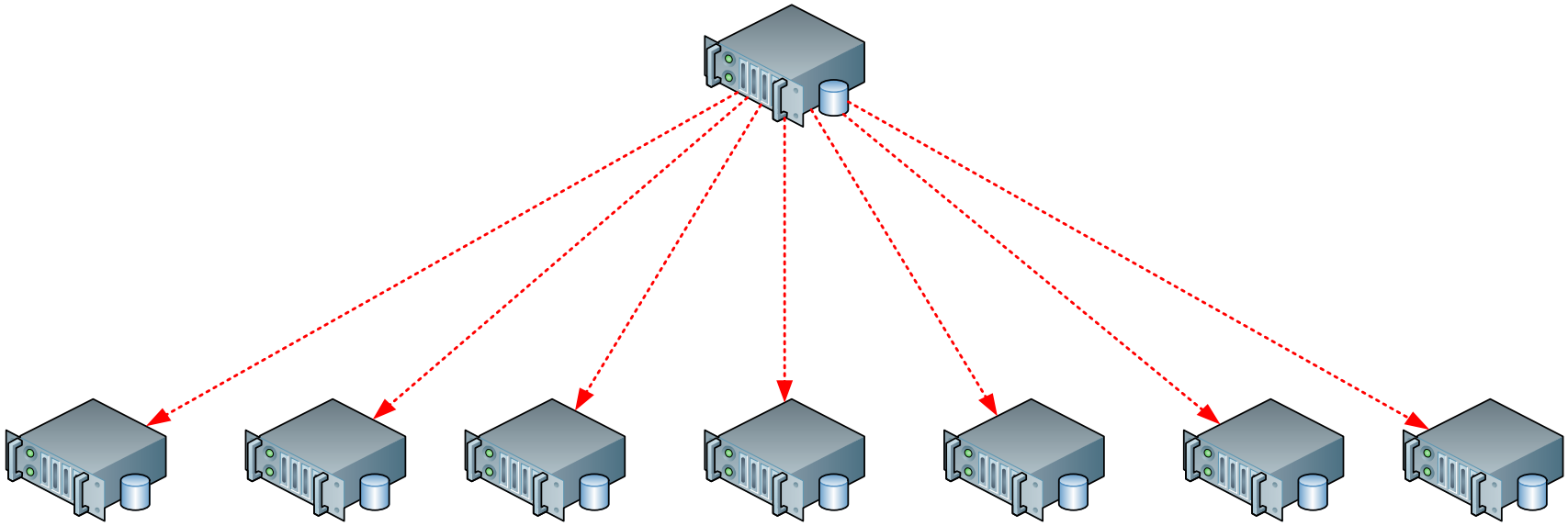
Persistent Connections pre-DRCP



The Distributed Database Problem: Replication

Replication is required to snapshot data from source database to client databases. For example, user data, including user_id, username, first name, main image, etc. are required on all distributed databases and applications. The main database is the master for the table, all other databases have a replicated read-only copy of this data.

- Materialized views and refresh groups are primary replication method



The Distributed Database Problem, continued

Database replication (materialized views)

- 20-25% overhead per client database (resource usage associated with keeping replicated tables up to date); additional overhead for masters that are sources of widely replicated data
- Minimum 1 minute lag between source and target(s) for fast refresh materialized view
- 788 registered materialized views on BP main database (example of a master)
- Releases are complicated by DDL modifications to “master” tables.
  - Mview logs can have to be recreated
  - Adding columns w/default values would fill up mview log
  - Multi-million row mviews that have to be rebuilt on 10-15 client databases, re-indexed, and constraints re-added takes a *long* time

The Old Solution

- TCP Connection Pooling @ Load Balancer
  - Reduces required number of processes by 90%
  - Still leaves us with over 125,000 persistent connections

- Web server clustering
  - Breaking web servers into discrete clusters reduces total number of persistent connections
  - Each cluster is built around an app or set of apps that share a set of databases
  - Each cluster connects to multiple schemas on one or more databases, instead of every webserver connecting to every schema on every database

The Old Solution: why not?

The Old Solution: why not?

- Extremely difficult to maintain:
  - Required constant rebalancing of web servers based on URL
  - Requires understanding of which databases each page/URL requires
  - Makes placement of new applications more difficult
  - Limits scalability of each cluster
  - Adds multiple single-points of failure -> downtime
  - Replication failures can cascade, increasing impact and length of downtime
  - Maintenance tasks multiplied by 15 (databases) – 25 (schemas)

- But…
  - It's cheap on the surface. We could use lots of little databases and Oracle Standard Edition (at the cost of administrative complexity).

History: Connection Pooling; What have we tried

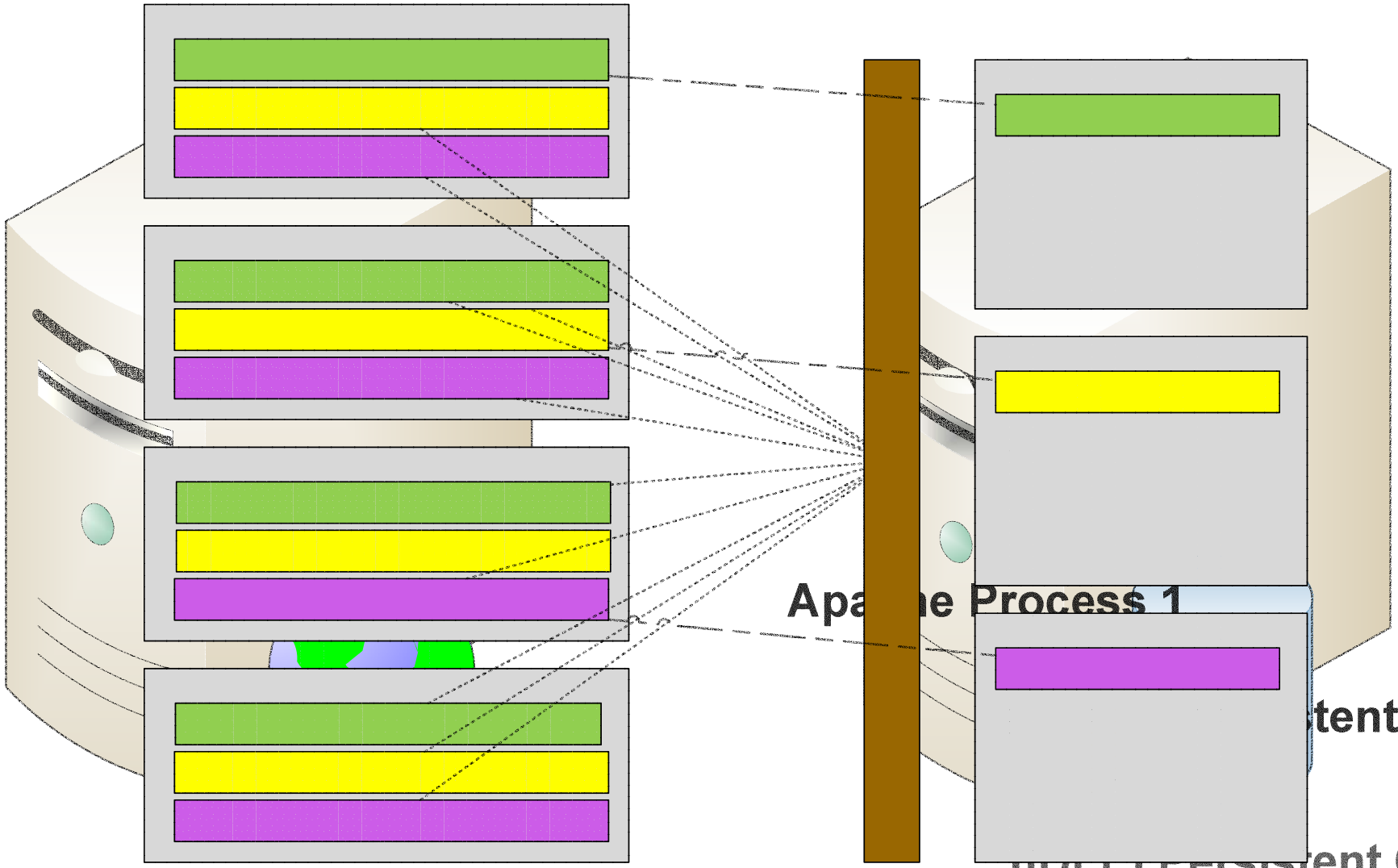
-Why did it take us so long to pool?

- PHP/Oracle: no middle tier
- What did we try?
  - Evaluation of 3rd party products
    - SQLRelay
      - Oracle layer antiquated; no updates to support key features
  - Internal development
    - Pros: complete control
    - Cons:
      - COMPLEX! Programming
      - Inability to “keep up” with Oracle new features
  - Shared Server (a.k.a. MTS) – Oracle 9i and 10g
    - Worked well for “small” sites
      - Memory savings were realized
    - CPU pegged at 100% for “large” sites
      - Code path was too long/complex to support high traffic

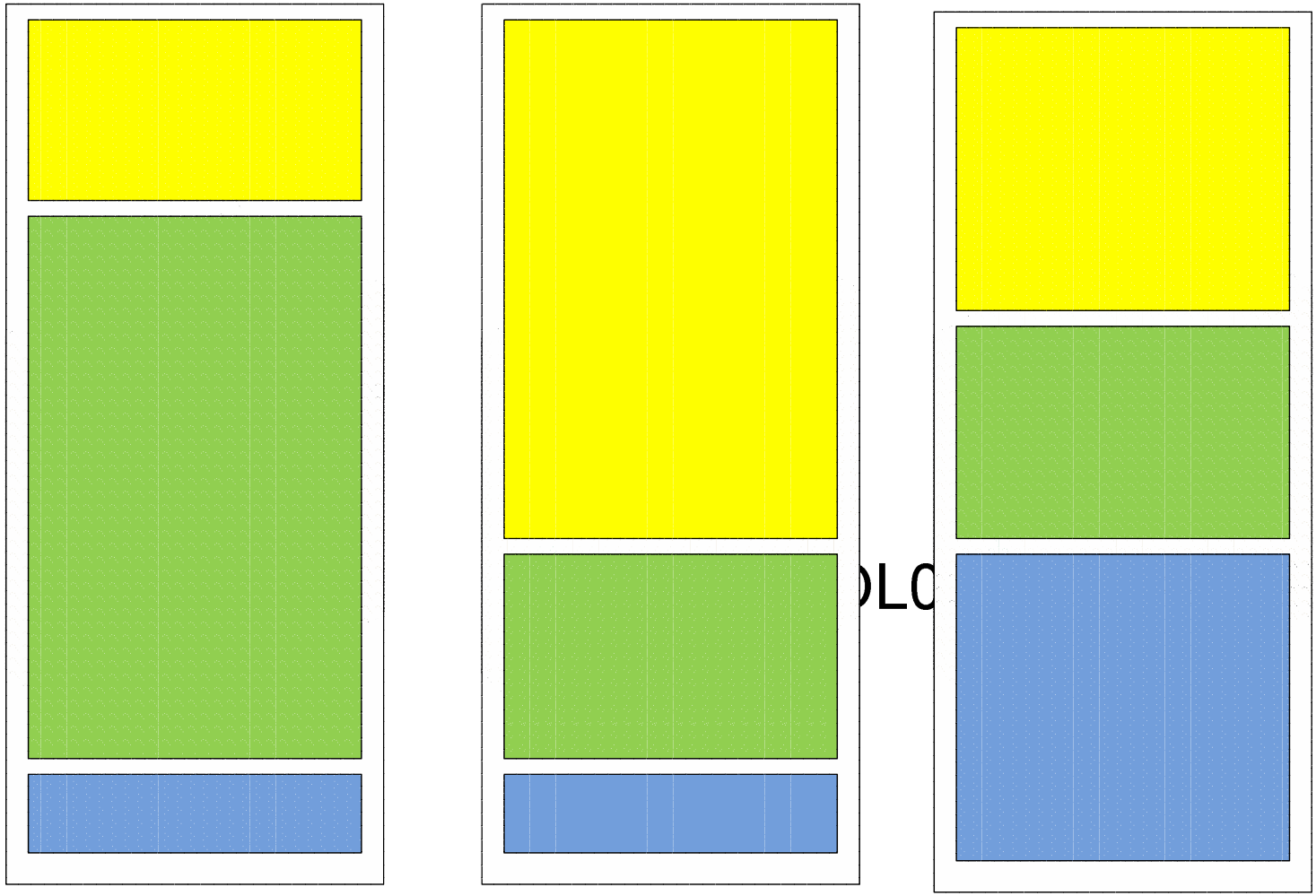
The New Solution

- Database connection and session pooling in conjunction w/11*g* Oracle RAC (on ASM)
  - Use DRCP with 11*g* to mitigate the memory wastage associated with persistent connection/shadow processes
    - Memory savings; only connection pool processes using shadow processes
    - No more web cluster management
  - Use RAC (w/DRCP) to ease administrative costs, development costs, and increase uptime
    - No more db replication management
    - One logical schema means simplified development
    - Rolling upgrades
    - Individual nodes can be lost without site outage
    - New nodes can be provisioned without downtime

The New Solution: Persistent Connections Post-DRCP



The New Solution



: PRE-DR

FREE MEMORY

Our Approach to Testing

-Functional Testing

- Install Oracle Instant Client on test web server
- Install PHP with OCI8 beta extension compiled in; deploy to test web server
- Install Oracle 11*g*R1 DB software to test DB machine
- Run unit tests (smoke test) from test web server, against 11g DB to ensure functional operation of core code base
- Run basic tests to open lots of connections from PHP CLI scripts

Testing (continued)

-Load and Scalability testing

- Upgrade 11*g*R1 on a “small” DB, to test scalability of DRCP
  - Increase number of web servers that connect to mgfind11 instance to prove “wide” scalability; that connection broker can handle many mostly unused connections in the context of connection pooling
- Upgrade BP canvas DB to 11*g*R1 with connection pooling to prove that DRCP can handle many idle connections w/a much higher transaction rate (exercise broker in higher transaction context)

Upgrading to 11G w/Connection Pooling

- Read Oracle documentation!
  - 11*g* upgrade guide and 11*g* minimum install requirements
  - Connection pooling documentation
  - White paper on DRCP
- Update TZ files on 10*g* instance pre-upgrade
- Modify kernel parameters (/etc/security/limits.conf) and “oracle” user environment to allow for more open file descriptors, etc.
- Dynamic service registration (change listener config)
  - DRCP Oracle processes use dynamic service registration to register with the listener; Explicit listener configurations (that can persist through an upgrade) will disable this and disallow the connection pool backend processes from registering with the listener
- Set compatible parameter to 11.1.0.0

Upgrading to PHP with DRCP

- nomenclature changes
  - oci_pconnect() with DRCP is now a pooled *session*
    - Sessions altered with ‘alter session’ can be reused by other scripts which didn’t alter the session

- Set oci8.connection_class based on workload
  - i.e. consider which sessions cursors you want cached, and how you want to reuse pooled sessions when determining which web servers use which connection class

Troubleshooting and collecting stats

- If you see waits in v$cpool_stats.num_waits increasing, it can indicate undersized maxsize for connection pool

- query V$cpool_stats to get general DRCP stats
- query V$cpool_cc_stats to get DRCP stats by connection class
- use netstat –t | grep ‘:1521’ to see connections to connection broker

Gotchas

Oracle RDBMS Patches and work-arounds:

- DRCP
  - Patch 6474441 for cursor leak
  - RAC related
    - Work around for CRS resetting maxconn_cbrok on restart/node eviction in 11*g* RAC environment
- 11*g* General
  - Patch 6677870 for “double-bind”
    - E.g. ”begin test_pkg.proc_1(:user_id); test_pkg.proc_2(:user_id); end;”
  - Work around for “create materialized view as select * from source_tab@source_site”
    - Use explicit column names

DRCP stats: BP production database

```
bp_prod_canvas@bptool02> @pt 'select * from v$cpool_stats';

POOL_NAME                       : SYS_DEFAULT_CONNECTION_POOL
NUM_OPEN_SERVERS                : 293
NUM_BUSY_SERVERS                : 261
NUM_AUTH_SERVERS                : 14
NUM_REQUESTS                    : 162349906
NUM_HITS                        : 162256770
NUM_MISSES                      : 93136
NUM_WAITS                       : 139925
WAIT_TIME                       : 0
CLIENT_REQ_TIMEOUTS             : 0
NUM_AUTHENTICATIONS             : 1626653
NUM_PURGED                      : 0
HISTORIC_MAX                    : 293
```