Caching Technologies for Web Applications

C. MOHAN
IBM Fellow

mohan@almaden.ibm.com
www.almaden.ibm.com/u/mohan/

IBM Almaden Research Center
650 Harry Road, K01/B1
San Jose, CA 95120, USA

Note: This is a **very preliminary** version of the VLDB 2001 tutorial slides. The final version will be available at www.almaden.ibm.com/u/mohan/Caching_VLDB2001.pdf

Agenda

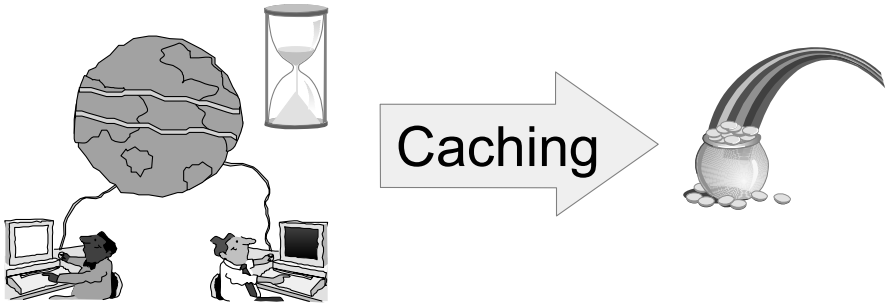
- ►Introduction
- ►Granularity of Caching and Location of Caches
- ►Content Delivery Services, Appliances, Edge Caching
- ►Dynamic, Personalized Content
- ►Database Caching Architectures and Issues
- ►Prototypes and Products
- ►Caching Case Studies

About the Speaker: Dr. C. Mohan joined IBM Almaden Research Center in 1981. He was named an IBM Fellow in 1997 for being recognized worldwide as a leading innovator in transaction management. He received the 1996 ACM SIGMOD Innovations Award. From IBM, he has received 1 Corporate and 8 Outstanding Innovation/Technical Achievement Awards. He has been an IBM Master Inventor with 33 patents. Mohan's research results are implemented in numerous IBM and non-IBM systems like DB2, MQSeries, Lotus Domino and S/390 Parallel Sysplex. He is the primary inventor of the ARIES family of recovery and locking methods, and the industry-standard Presumed Abort commit protocol. At VLDB'99, he was honored with the 10 Year Best Paper Award for the widespread commercial and research impact of the ARIES algorithms. He has been an editor of VLDB Journal, and Journal of Distributed and Parallel Databases. Currently, Mohan is a member of the IBM Application Integration Middleware (AIM) Architecture Board and is leading a project on database caching in the context of DB2 and WebSphere.

Motivation

World Wide WAIT!

Happy Customers!



Caching is widely used for improving performance in many contexts (e.g., processor caches in hardware and buffer pools in DBMSs)

Where and what to cache in web context?
Many places and many types of objects!

Focus of tutorial: **Transactional** applications, not internet search, etc.

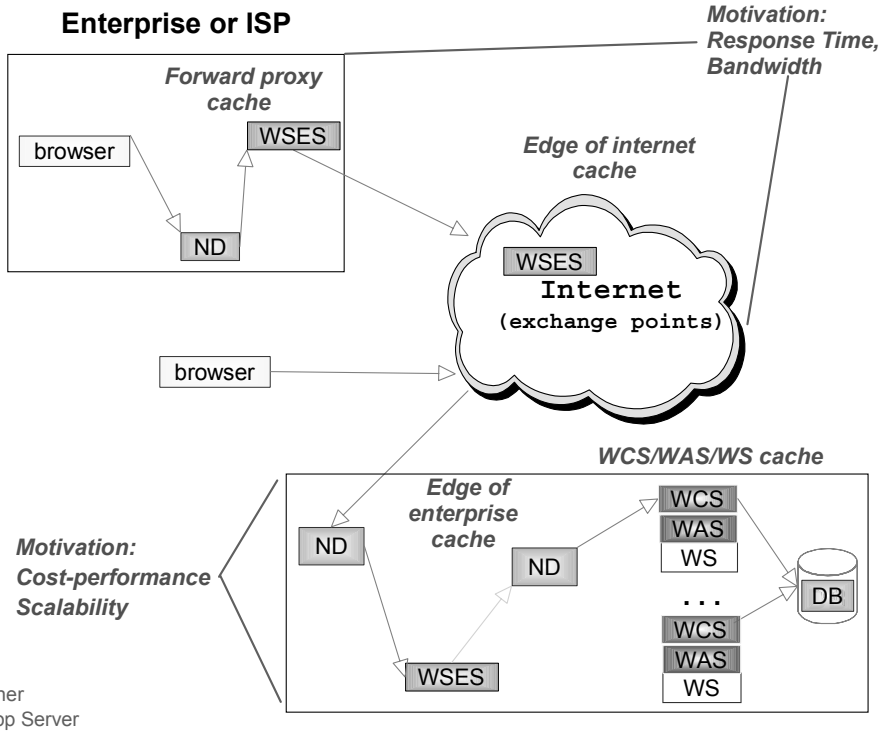
What, Where, When and How

Considerations for caching in web context are:

- What, when and where to cache
- Granularity of caching: web page, fragment of a page, servlet execution result, SQL query result, relations, ...
- Location of cache: client, proxy server, edge-of-network server, ISP, edge-of-enterprise server, application server, web server, DBMS
- Caching and invalidation policies: application transparency, push vs. pull, time-to-live, triggers, log sniffing, ...
- Enabling cache exploitation: routing, failover, authentication, authorization, accounting, ...
- Tools for cache performance monitoring and analysis

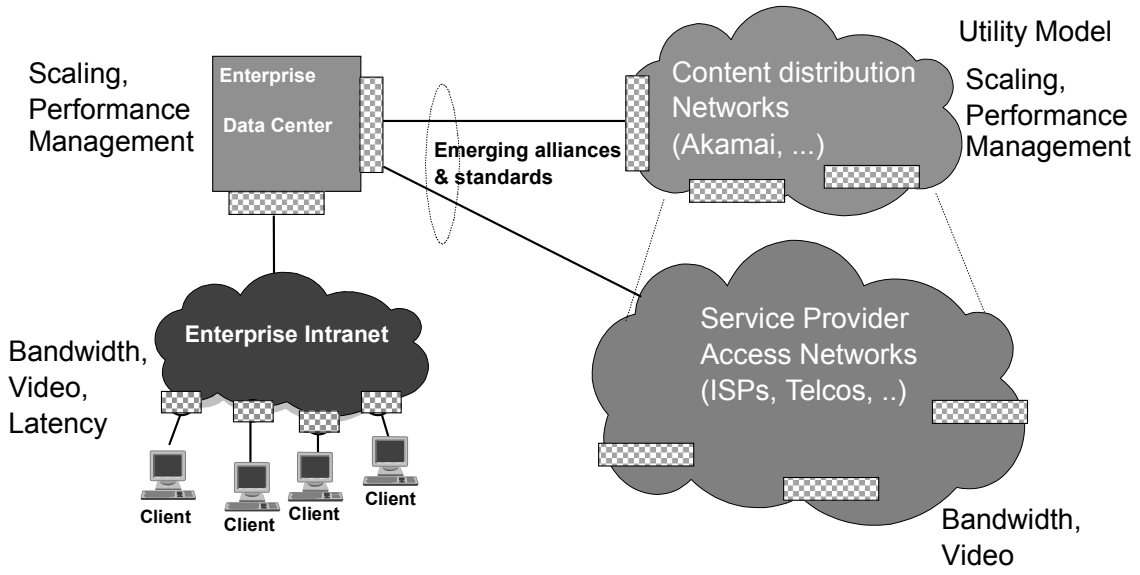
Related DB Technologies: replication, materialized views, mediator systems, client server DBMSs, buffer management, main memory DBMSs, query rewriting/optimization, ...

Common Points of Caching (Non-Database)



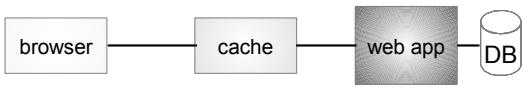
DB: Database
ND: Network Dispatcher
WAS: WebSphere App Server
WCS: WebSphere Commerce Suite
WS: Web Server
WSES: WebSphere Edge Server

Edge Serving



Cache Models

Front-End Cache

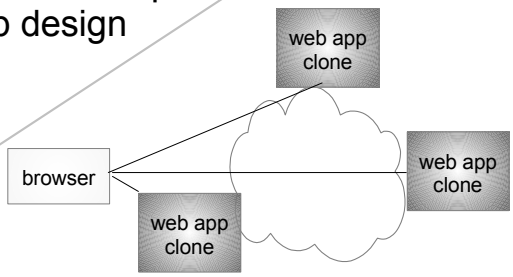


- caches data + markup
- cache can be app independent
- static pages easily cached
- dynamic pages require app support

Data Cache

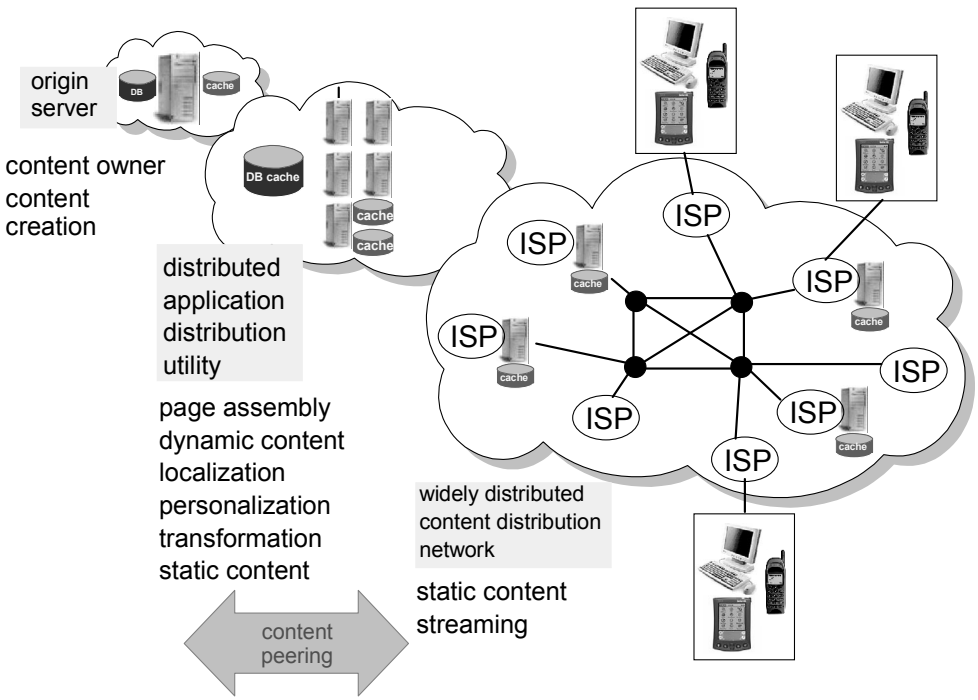
- caches data
- effectiveness of cache usually dependent upon app design

Distributed App

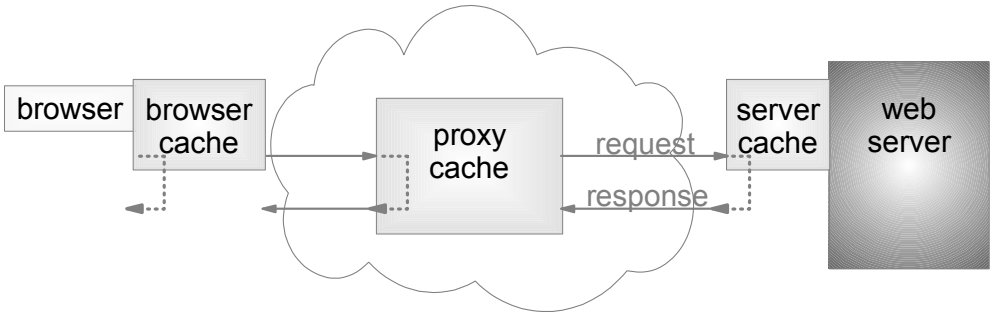


- multiple app copies distributed around net
- turns caching into content management problem
- distribution concept built into app (Zembu is changing this!)

Tiers of Content Serving

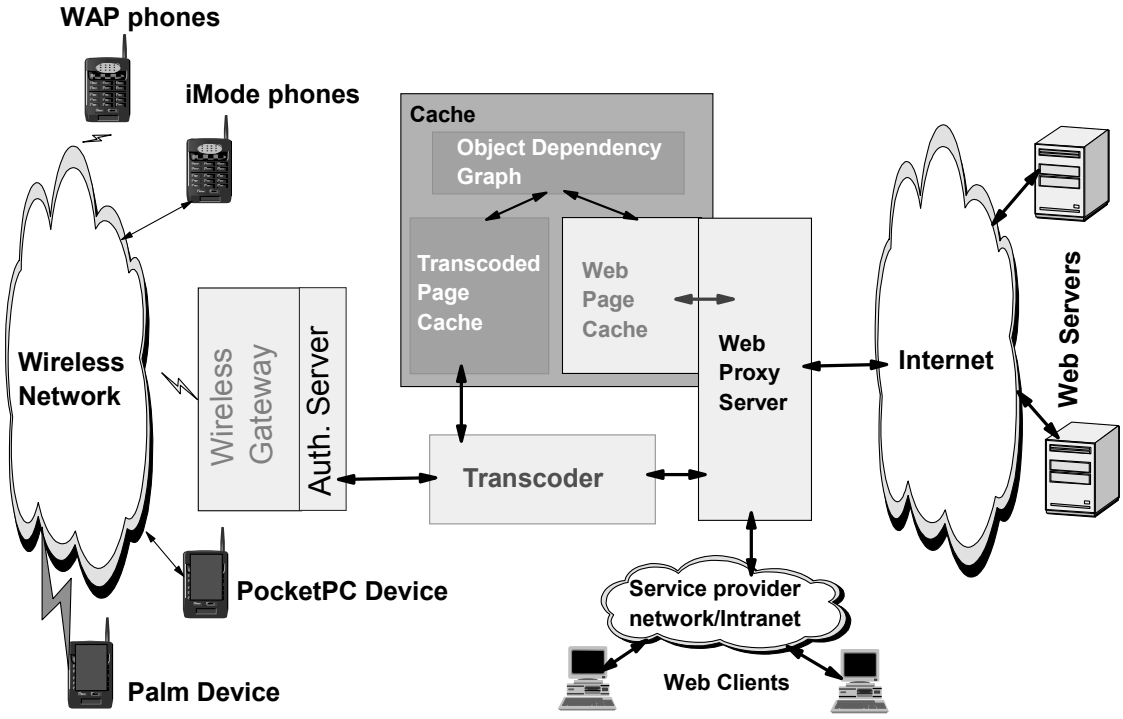


HTTP Caching Today



- Multiple caches - anywhere between browser and server
- HTTP headers control
  - whether or not a page can be cached
  - cache expiration time (Time To Live - TTL)
- Full pages and images can be cached
  - unable to cache HTML fragments

Multi-Device and Mobile Web Caching

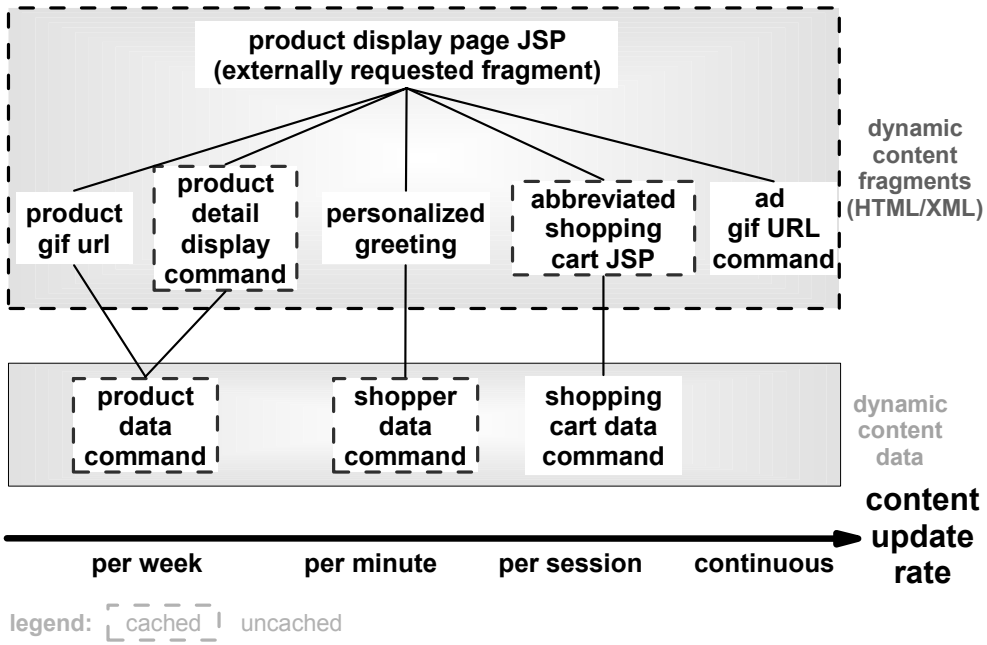


Specialized Caching Vendors/Software

- Content delivery service providers: Akamai, Digital Island, Content Bridge
  - Techniques for request rerouting - URL rewriting, DNS redirection
- Cache appliances: Network Appliance, CacheFlow, Cisco Cache Engine, InfoLibria DynaCache
- Edge of network caching software: IBM WebSphere Edge Server

Caching HTML Fragments

"When part of a page is too volatile to cache, the rest of the page can still be cached."



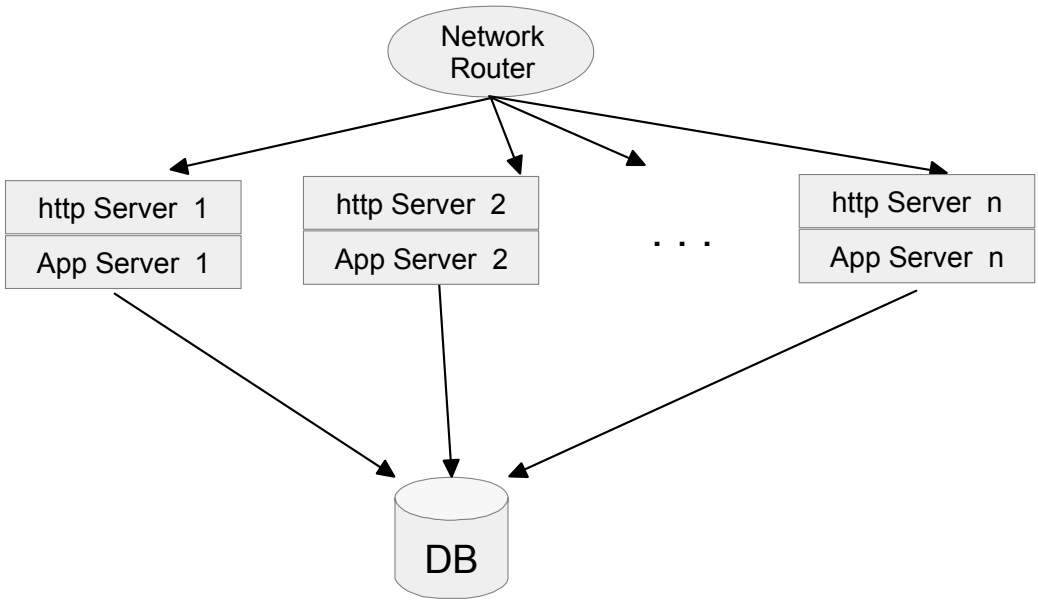
Fragment Caching Goals

- Achieve benefits of cache for personalized pages
  - Improved price/performance
  - Improved response time latency
- Reduce cache storage requirements
  - By sharing common fragments among multiple pages
- Support contracts & member groups in commerce apps
- Move cache into the network to multiply above benefits

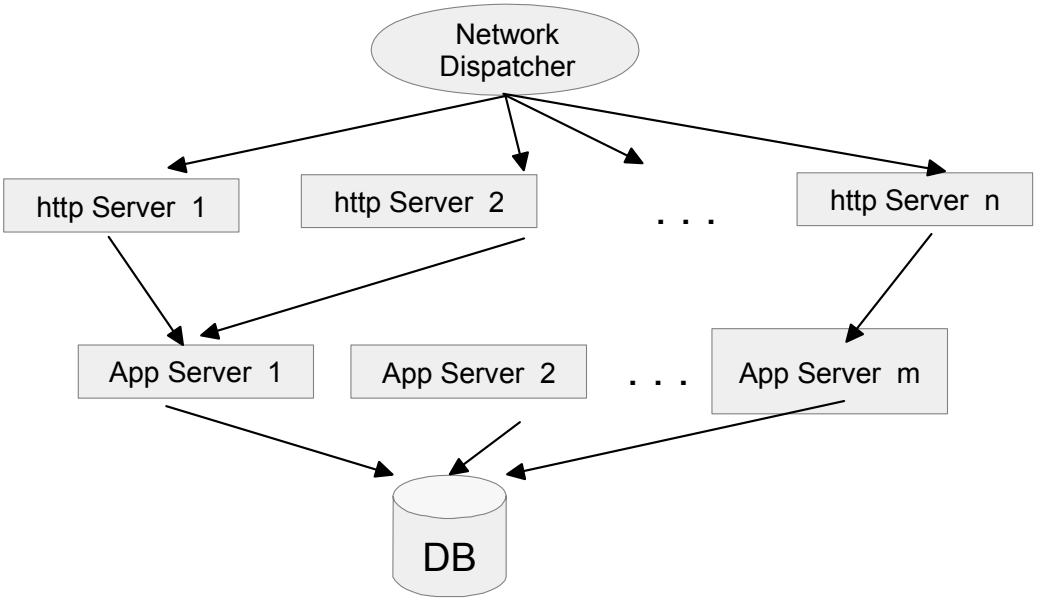
Database Caching

- Corporate data: backbone of eCommerce apps
  - Static data cached at app level (e.g., catalogs in WCS)
- Current Strategies
  - App-aware caching model - OK for app-specific data (web pages, images, etc.)
  - Replication: cannot easily track usage patterns, not dynamic enough to adapt to changing access patterns
- Caching of database data
  - Scalability: offload work from backend database server
  - Reduced response time: caching at an edge server
  - Reduced cost of ownership (e.g., use less reliable, cheaper machines for caching)
  - Improved throughput, congestion control, availability, QoS

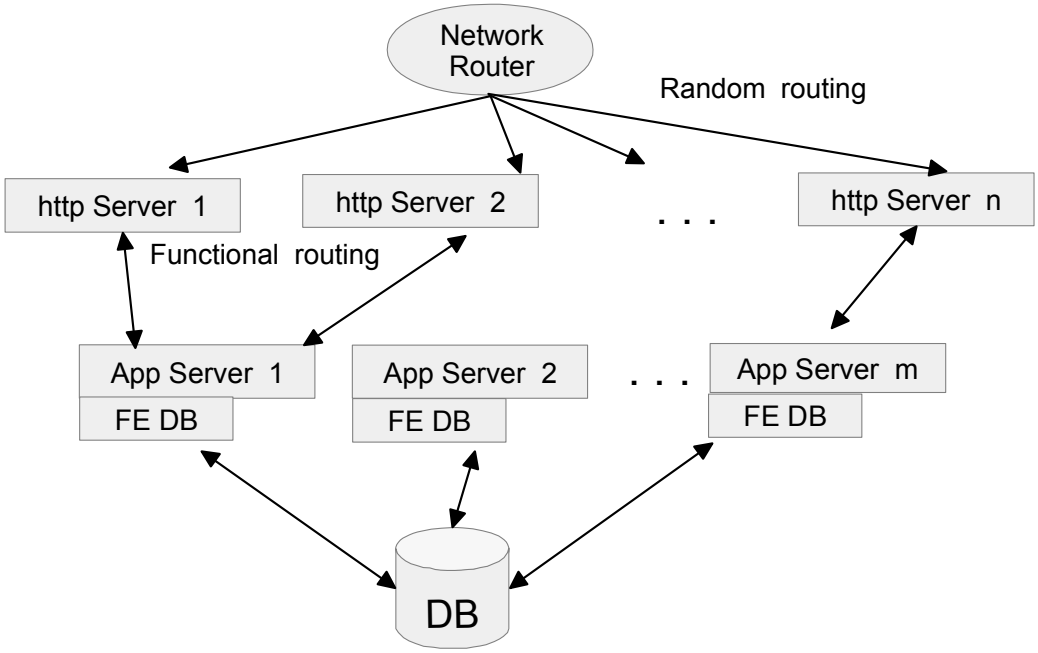
Classical Web Setup - 3 tier



Classical Web Setup - 4 tier



Web Setup with Data Caching



Frontend DB could be same brand as backend DB or a different one

Types of Caching

	what is cached	where it is cached	programming model	how to invalidate
fragment cache	*fragment (HTML or XML)*	*application server or edge server*	*JSP/servlet or tags*	*time limit or explicit*
command cache	*query result or fragment*	*application server or edge server*	*command*	*time limit or explicit*
query result cache (Wisconsin, Watson)	*query result*	*application server*	*SQL*	*time limit or explicit*
Database cache (Oracle, TimesTen, Almaden)	*base data*	*application server host (separate process)*	*SQL*	*automatic (replication)*
Database cache + application (Ejasent, Zembu)	*app + base data*	*edge server*	*various*	*automatic*

Middle-tier Data Model Requirements

- App's SQL ***should not have to change***
- App's db schema ***should not have to change***
- Must support ***failover*** of nodes
- Must support ***reasonable update semantics***
- Must support dynamic addition of nodes
- QoS: Predictable update propagation latencies
- Aim to process queries in middle-tier

Middle-tier Data Model Choices

- ***Cloned:*** Each table is identical to a backend table
  - Pros
    - DDL definition is easy
    - Every query can be satisfied anywhere
  - Cons
    - Updates need to reach multiple nodes
    - Adding nodes involves copying lot of data

- ***Subset:*** Each table is a proper subset of a backend table
  - Pros
    - Updates can be handled in minimal sites
    - Performance: Smaller databases in middle-tier nodes
  - Cons
    - Complex Routing: integrate with edge server
    - Complexity in DDL definition and query processing
    - Updates logic complicated - need to know who owns records being changed

Materialized Views

- CREATE SUMMARY TABLE west_coast_emp AS
  (SELECT * FROM employee
  WHERE employee.state IN ('CA', 'WA', 'OR'))
- Materialized views called ASTs (Automatic Summary Tables) in DB2
- Query rewrite routes query to materialized view instead of base table
- Refresh: Incremental/Defered
  - If deferred refresh, materialized view used only if user says out-of-date data is okay (refresh age)

Tables and queries

- Consider tables (TPC-W style)
  - Customer (cid, cname, ...)
  - Order (cid, oid, otime, oprice, ...)
  - OrderLine (oid, olid, itemid, ...)
- Queries:
  - select * from customer where cid = 123
  - select * from order where cid = 123
  - select * from order where oid = 345
  - select * from orderline where oid = 345
  - select * from orderline where olid = 567
  - select orderline.* from order, orderline
    where orderline.oid = order.oid and order.cid = 123

Database Caching Design Goals

- Given
  - Backend Database Schema
  - Web Application + workload
    - URLs and corresponding SQL queries
- Need to generate
  - Middle-tier data model
    - DDL for Nicknames, Cached Tables, Views
    - Data partitioning scheme
- Need to manage
  - Workload manager routing based on partitions
  - Adding and removing nodes dynamically

Cache Refresh

- Automatic
  - time-driven
  - immediate (synchronous, asynchronous)
- On-demand
- Refresh brings new content or only invalidates old cached content

Updates - Push Vs Pull

- Push advantages
  - reduced response time for first hit
  - overwrite has less total path length than invalidation + pull
- Pull advantages
  - everything does not have to fit in the cache
    - not in cache until accessed
    - only hottest stay in cache long term
  - user A's personalized info can be cached close to user A without being cached everywhere
  - easy to get context required to execute JSP
    - a real request is always underway when executing a

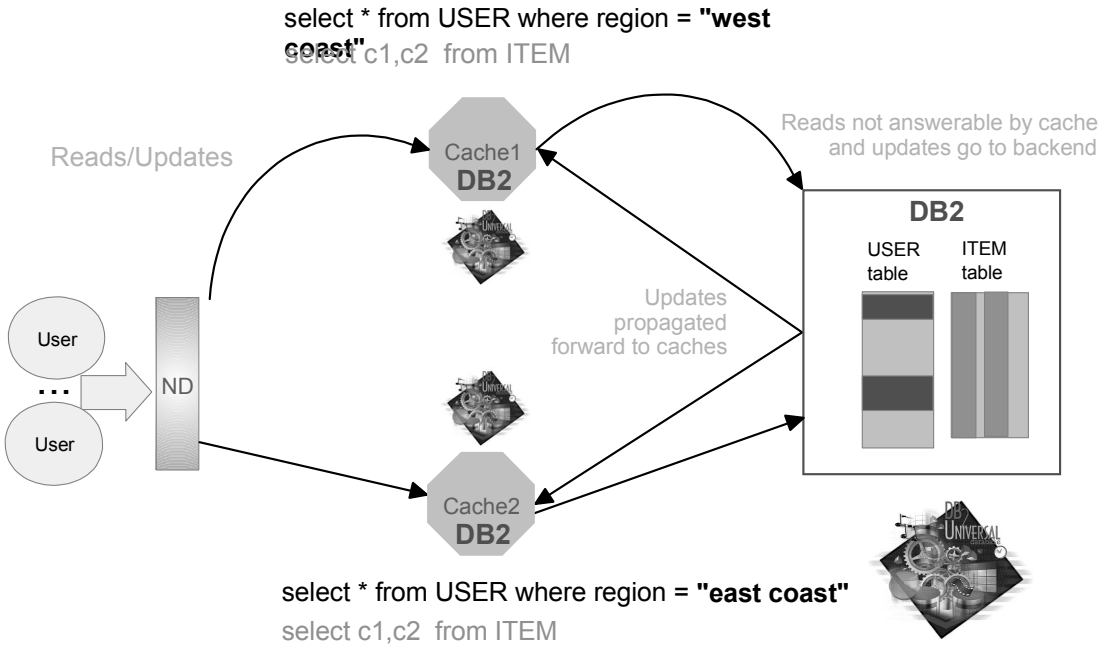
Update Handling

- Where to perform the updates first
  - Frontend only
  - Backend only
  - In both places (two-phase commit - cost and availability issues)
- Use asynchronous propagation if update performed first in only one place
- Semantic problems
  - Users not seeing their own updates
  - How updates are replayed on other copies (problem with logical redo if identity columns are involved)

Data Caching Choices

- If cache in database process, app server will incur costs of
  - process boundary crossing
  - data conversions
- Alternative is to cache in app server process itself data in
  - relational format (JDBC query results) for servlets, session/entity beans
  - Java object format for entity beans
- App server would have to manage cache coherency, especially in cluster environment!

IBM Almaden's DBCache Project



Components of DBCache

- Seamless access to front-end data and backend data
  - DB2 Federated nickname support
- DBMS is told front-end data is subset of backend data
  - Conditional sections in plan to route to backend DB2 if needed data not in cache - allows more dynamism in what is cached
- Conduit for propagating changes to frontend
  - DPropR

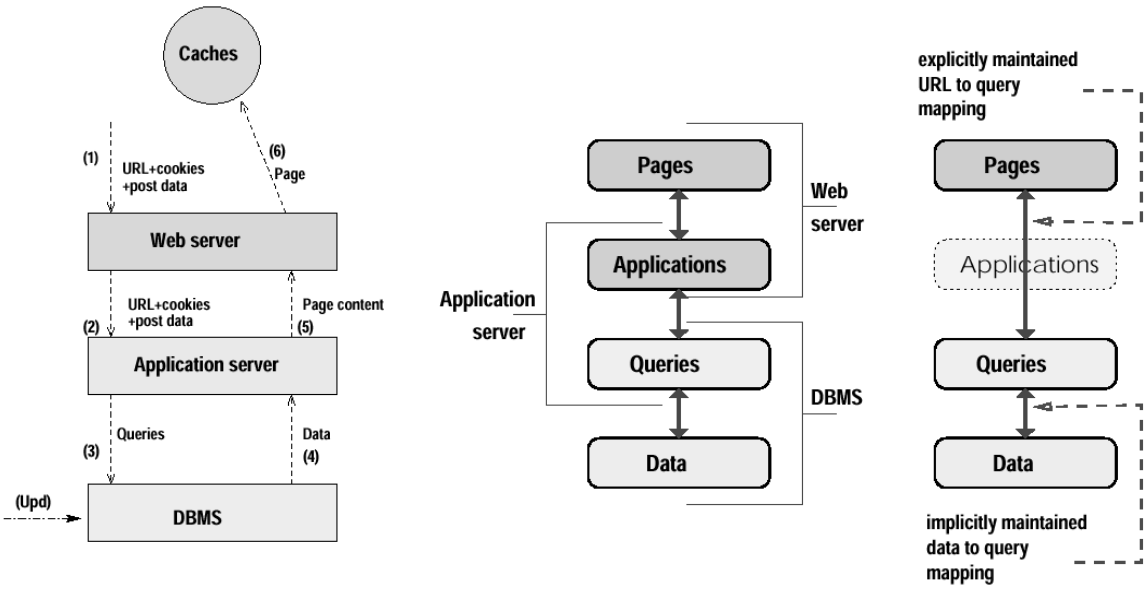
Challenges

- Describing cache contents
  - Expand on capability of materialized views
- Replication subscriptions: How to do dynamic modification?
- Intelligent routing by WebSphere Edge Server
- Query workload analysis to determine view definitions
- Tracking a changing workload using view as a cache granule
- Handling of updates & associated read semantics: Choose from apply update to FE/BE only, both FE & BE
- Cache synchronization
- Query optimization
- Failure handling
- DBA tools

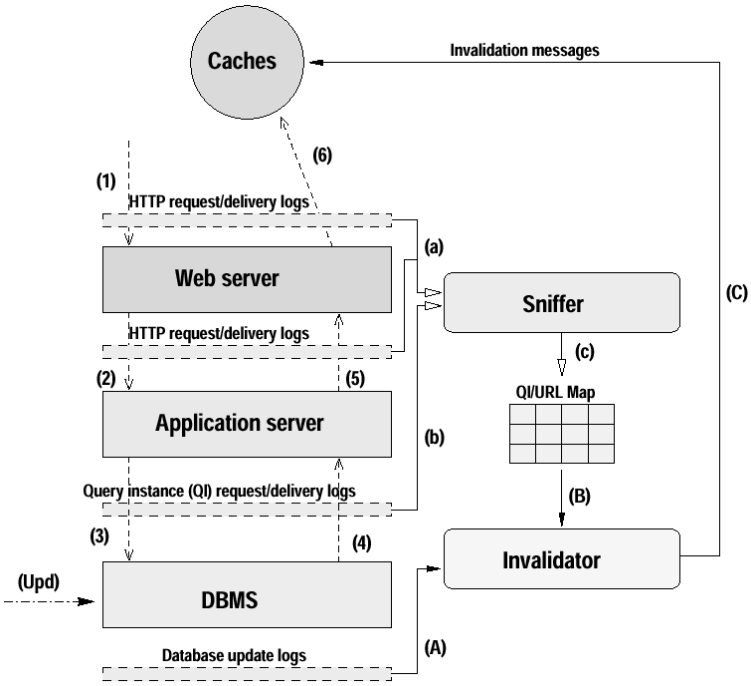
Query Result Caching

- Cache could exist within or outside DBMS - could integrate with JDBC driver
- Results Tracking
  - Individual results kept separately
  - Results combined to avoid duplicate subsets
- Cache used to answer
  - only repeated (exact match) queries: just return bits in bucket, no need for fancy query engine
  - any query for which answer exists in cache (subset or union of earlier queries): need query processing capabilities
- Full exploitation of outside-DBMS cache requires
  - replication of significant DB query handling functions
  - managing invalidation of results is much more difficult
  - modified queries to be sent to backend DB to retrieve all columns and all qualifying rows

NEC Work



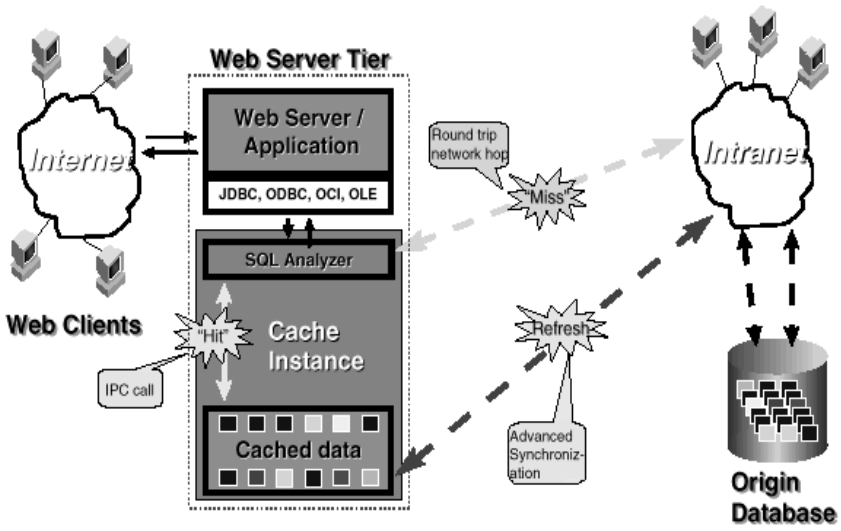
NEC Architecture



Database Caching Products

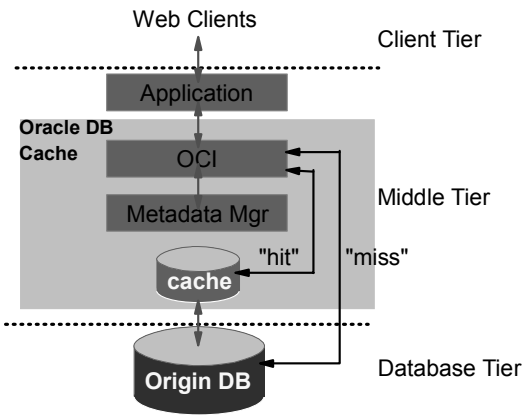
- A number of companies are currently active in this space
- Not really a new area!
  - OODBMSs did non-persistent caching on clients
  - ObjectStore did caching beyond transaction commit by call-back locking
- Scope being extended to edge of the network
- Main memory technologies being exploited in some cases

Oracle 9i IAS - Relational Cache



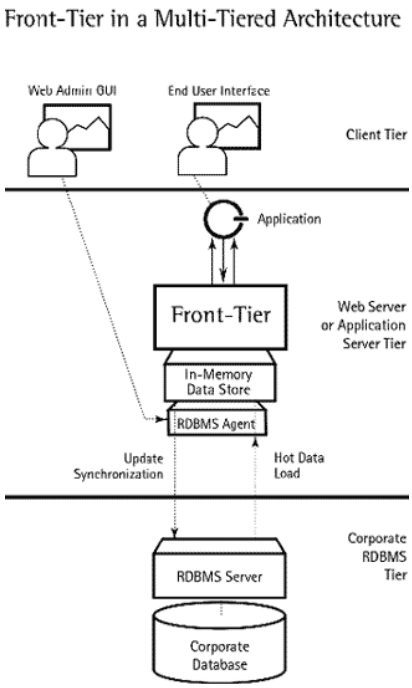
Oracle 9i IAS - Relational Cache

- You can cache only entire contents of a table and not a subset of a table
- Queries processed entirely using cache or entirely sent to backend; updates always sent to backend; Only one backend DBMS per app server
- Good tooling for set up and monitoring cache performance
- Allows per-statement turning on of whether the cached data can be used or not
- Access to cached data can be done at a global level or on a per-connection level (latter requires recompilation of app)
- Read from cache after update to origin by same transaction will return old value
- PL/SQL objects (packages, procedures, functions) that contain read-only requests and that refer only to cached tables can also be cached



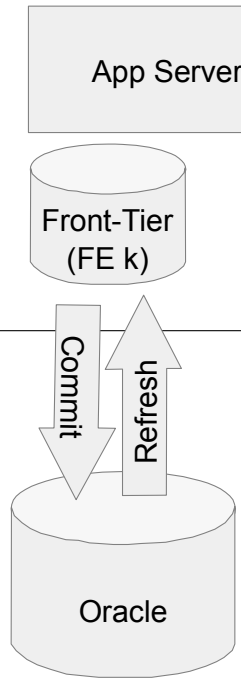
TimesTen

- Original HP Lab main-memory project spun out as startup
- MMDB product TimesTen
- Oracle data caching product Front-Tier
- Cache Group defines tables to be cached (with join, selection criteria)
- Cache updates propagated on commit or user command
- Options for enabling/disabling logging, durability of log (in-memory or disk-based), durability of data
- App designer control over cache loading, refresh and flushing



TimesTen's Front-Tier

Cache Group concept to cache related records from different tables

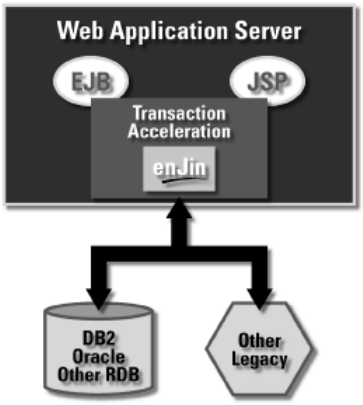


Pros:
- Front-Tier - optimized for speed
- Sub-second repl FEx - FEy

Cons:
- App is aware of Front-Tier
- Two SQL Dialects
- Updates flushed only on Commit
- Batch refresh only

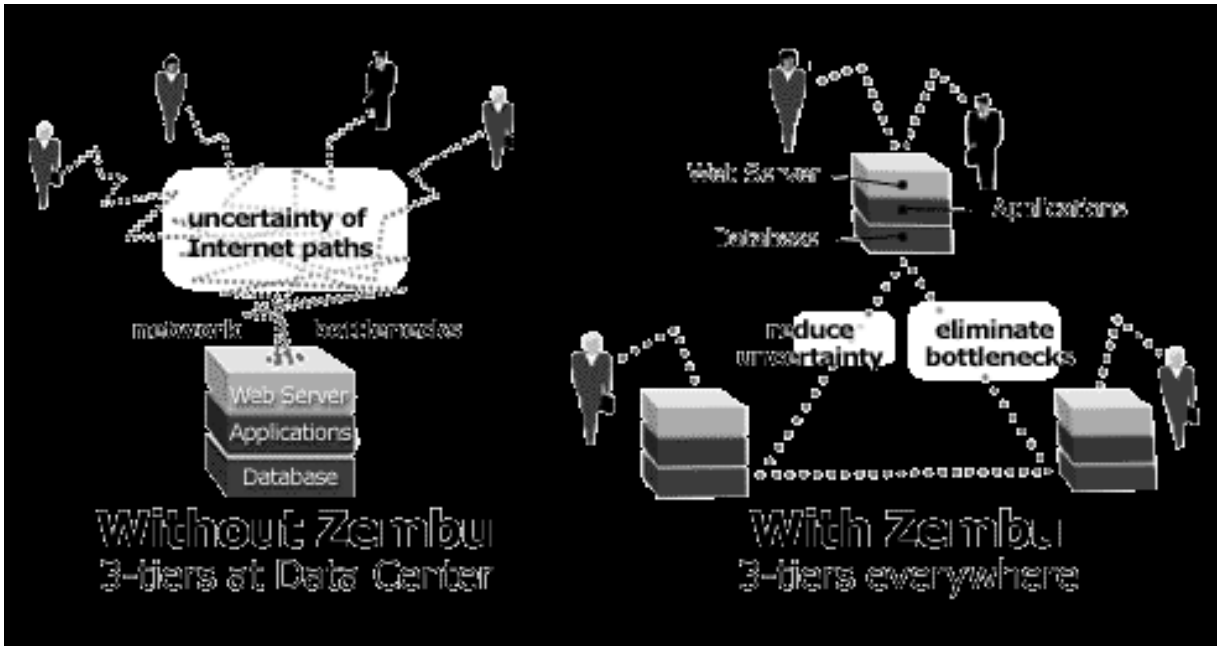
Versant's enJin

- Provides Java and EJB persistence
- Works with WebSphere and WebLogic
- Propagates updates to RDBMSs
- Supports XML



Zembu

- Startup doing "Akamai for apps" for dynamic content - 2/99



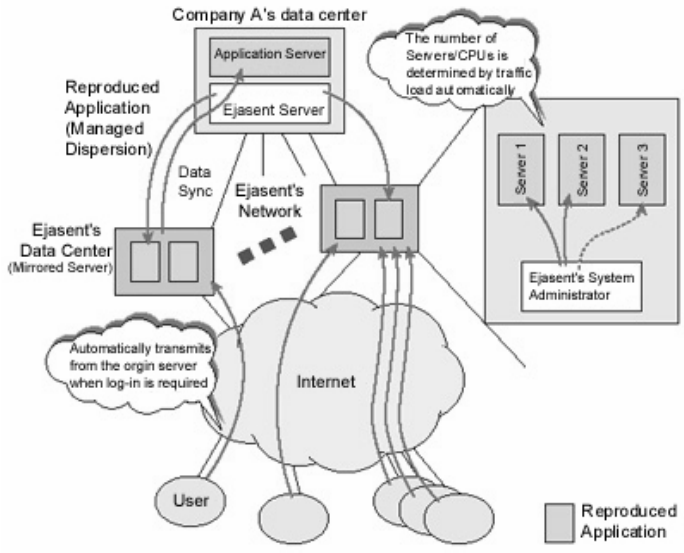
Zembu

First a hosted service - Now a shrink-wrapped product!

- Distributed Publishing Manager
  - Distributes centralized app stack to many sites
  - Integrates with existing content management and development products
- Distributed Infrastructure Manager
  - Configures and manages network
  - Optimizes end-user routing for optimal performance
- Distributed Data Manager
  - Coordinates distributed data and supports replication
  - Synchronizes changes of concurrent users
  - Migrates data to where it is used most

Ejasent

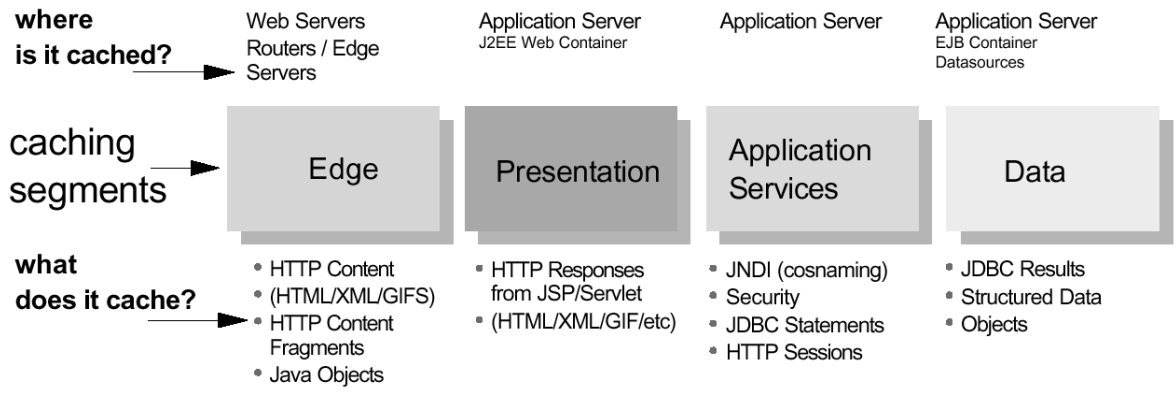
- Startup owns and operates a distributed, interactive service net
- Users in real time allocate additional resources
- BEA and Broadvision Solaris apps - snapshot of binary images captured and run on Ejasent servers



Case Studies

- ►IBM WebSphere
- ►Olympics
- ►eBay

WebSphere Caching Landscape

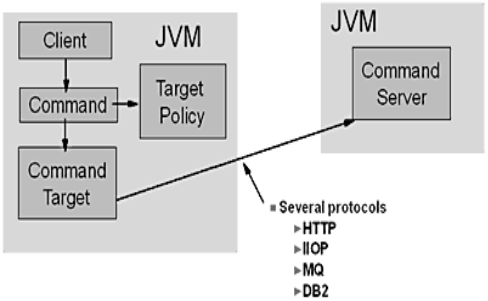


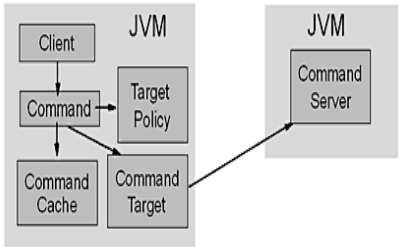
Dynacache (WebSphere V4.0)

- ►Application developers/assemblers control how fragments are cached using XML cache descriptor
- ►Define rules based on servlet, URI, request attribute/parameter/session/cookie
- ►Rule/time-based, and programmatic techniques for invalidating cache entries
- ►Can control external caches, e.g. WebSphere Edge Server, FRCA

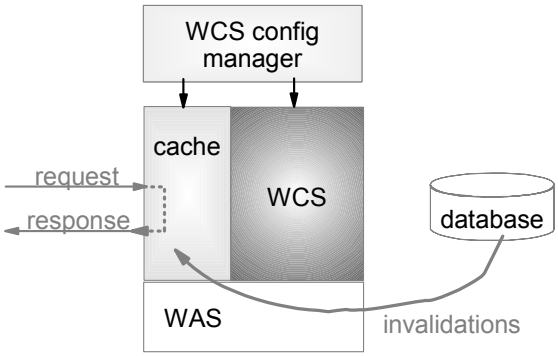
WebSphere Commands and Caching

- Command: a JavaBean that represents a particular operation and can be serialized
- Commands isolate apps using them from the implementation of the services they reach through them
- The context needed to execute the command is carried in the command in the form of its setable properties
- Results of executing a command are carried in the command in the form of getable properties

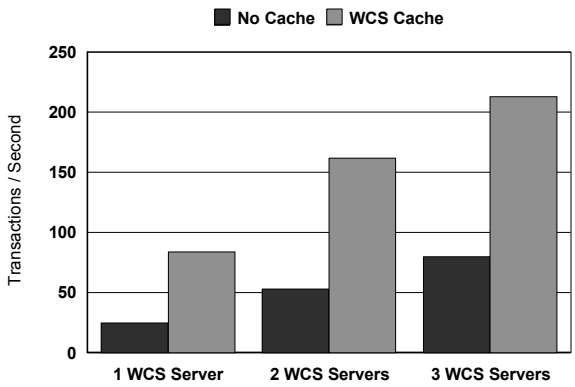




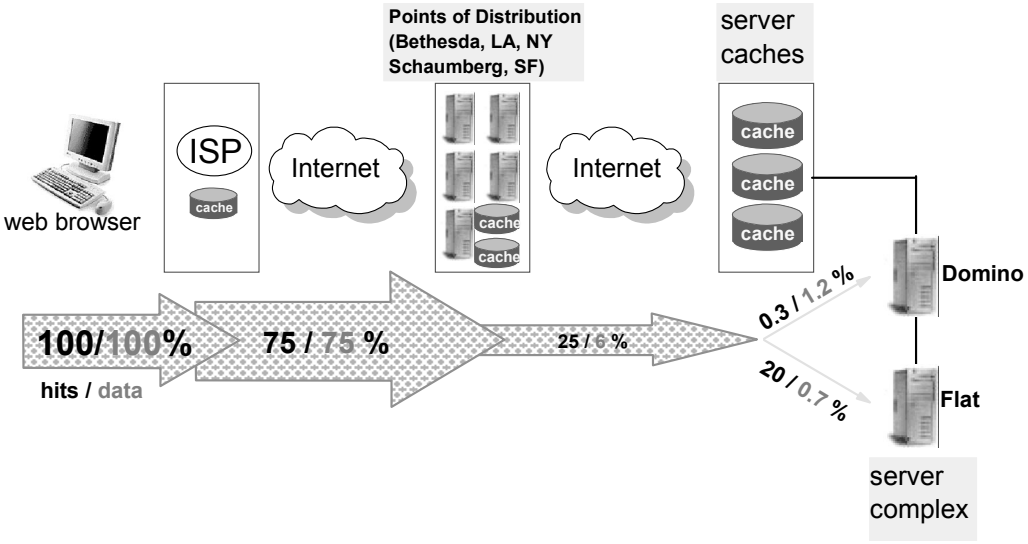
WebSphere Commerce Suite (WCS) Cache



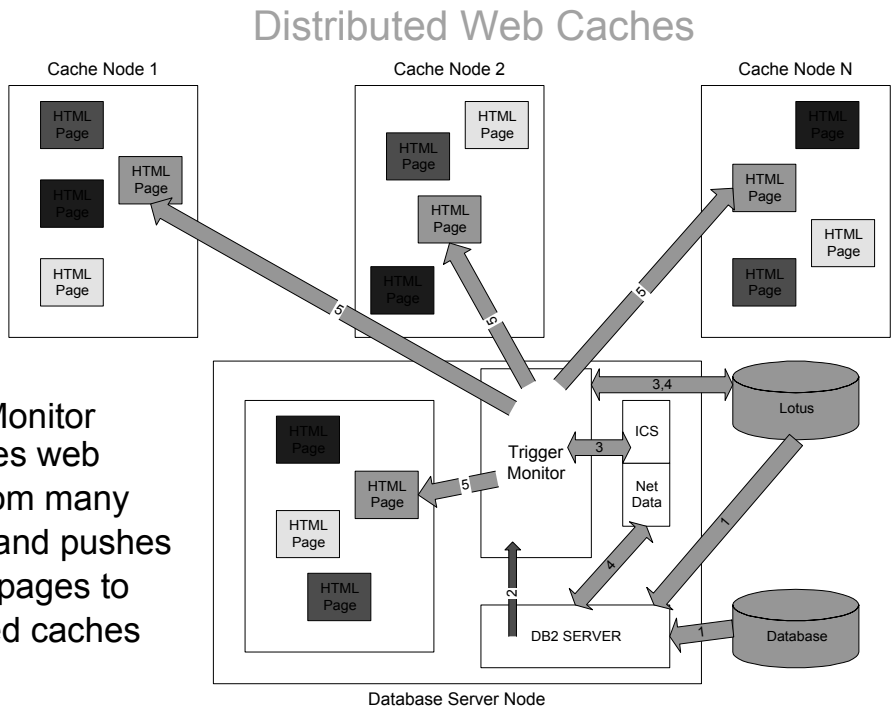
- Typically used to cache catalog data
- Admin can influence what gets cached
- Right now, personalization turns off caching



Effect of Caching at Olympics



Dynamic Data Caching for Olympics



Trigger Monitor assembles web pages from many sources and pushes updated pages to distributed caches

eBay

- Backend: Sun machines running Oracle
- Front-ends: Intel boxes running some specialized DBMS
- Data cached in FEs refreshed every 30 minutes
- App level routing of queries to FE vs BE

As of October 2000

- 18.9M registered users
- Items in 4,500 categories
- Millions of items for sale
- Users add 600K new items daily
- 4M page views per month
- Daily reach topped 16.2%, # of unique visitors on an average daily basis is 2.1M
- Average usage per month by a user is 119.6 minutes

Challenging Open Issues

- Access control at the edge
- Standardized naming
- Session state tracking and failover support
- Cache synchronization
- Performance monitoring and tuning DBA tools
- Load balancing or cache-intelligent URL routing
- Describing cache contents
- More sophisticated query optimization criteria
- Efficient relational to Java object mapping
- XML data caching

Edge Side Includes (ESI)

- Markup language for defining web page components for dynamic assembly and delivery at network edge
- Specifications released for ESI language, Edge architecture, Invalidation protocol and JESI tag library
- Coauthored by Akamai, ATG, BEA Systems, Circadence, Digital Island, IBM, Interwoven, Oracle, and Vignette
- http://www.esi.org

References

- "Distributed Application Platform", White Paper, Zembu Labs, February 2001.
- "Oracle9i Application Server Database Cache", Technical White Paper, Oracle, http://otn.oracle.com/products/ias/pdf/db_cache_twp.pdf, October 2000.
- "ESI Language Specification 1.0", http://www.esi.org/language_spec_1-0.html, 2001.
- Internet Caching Resource Center, http://www.caching.com/
- Akamai EdgeScape White Paper, Version 1.1
- "Cutting the Costs of Personalization with Dynamic Content Caching", Aberdeen Group, March 2001.
- "Automated Script Analyzer and Processor (ASAP)", Technology Brief, Chutney Technologies.
- "Dynamic Content Acceleration: A Caching Solution to Enable Scalable Dynamic Web Page Generation", White Paper, Chutney Technologies, May 2001.
- "A Middleware System Which Intelligently Caches Query Results", Louis Degenaro, Arun Iyengar, Ilya Lipkind, Isabelle Rouvellou. Proc. ACM/IFIP Middleware 2000, Palisades, April 2000, Springer-Verlag.
- "High-Performance Web Site Design Techniques", Arun Iyengar, Jim Challenger, Daniel Dias, Paul Dantzig. IEEE Internet Computing, March/April 2000.
- "A Publishing System for Efficiently Creating Dynamic Web Data", Jim Challenger, Arun Iyengar, Karen Witting, Cameron Ferstat, Paul Reed. Proc. IEEE INFOCOM, March 2000.

References

- Brian D. Davison's Web Caching and Content Delivery Resources, http://www.web-caching.com/
- "Enabling Dynamic Content Caching for Database-Driven Web Sites", K. Selçuk Candan, Wen-Syan Li, Qiong Luo, Wang-Pin Hsiung, Divyakant Agrawal. Proc. ACM SIGMOD International Conference on Management of Data, Santa Barbara, May 2001.
- "Scaling up e-business applications with caching", Mike Conner, George Copeland, Greg Flurry. DeveloperToolbox Magazine, August 2000. http://service2.boulder.ibm.com/devtools/news0800/art7.htm
- "Form-Based Proxy Caching for Database-Backed Web Sites", Qiong Luo, Jeffrey Naughton. Proc. VLDB 2001, Rome, September 2001.
- "Versant enJin for IBM WebSphere", http://www.versant.com/products/enjin/whitepapers/WP_9908-rev0103r4W.pdf