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 the final version of the slides and is available at www.almaden.ibm.com/u/mohan/Caching_VLDB2001.pdf
12 September 2001

Agenda

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

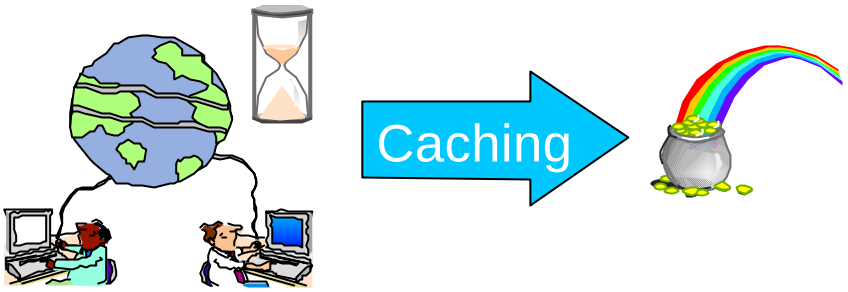
NOT a survey of all known algorithms/projects!!

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 in recognition of his innovative contributions to the development and use of database systems. 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, Microsoft SQLServer 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 caching points and many types of objects!

Our focus: **Transactional/Database** apps, **not** internet search, etc.

e-Business Application Characteristics

Study by IBM's Conner, Copeland, Flurry

- Large # of registered and online users
- Load dominated by inquires that grow without natural limits
- More tolerance for timeliness of data
- 24x365: an increasingly strong goal
- e-Business app's users are its customers
- Multi-tier
- Multiple channels of access are the norm for many apps

e-Business Scaling Principles

- Move interactions as close to user as possible
- Caching: key technique to reduce costs and improve response time
- Spread load over an expandable # of servers
- Segment load by tiers and within tiers make computations in a server more homogeneous
- Manageability, security and availability are critical factors in all design decisions

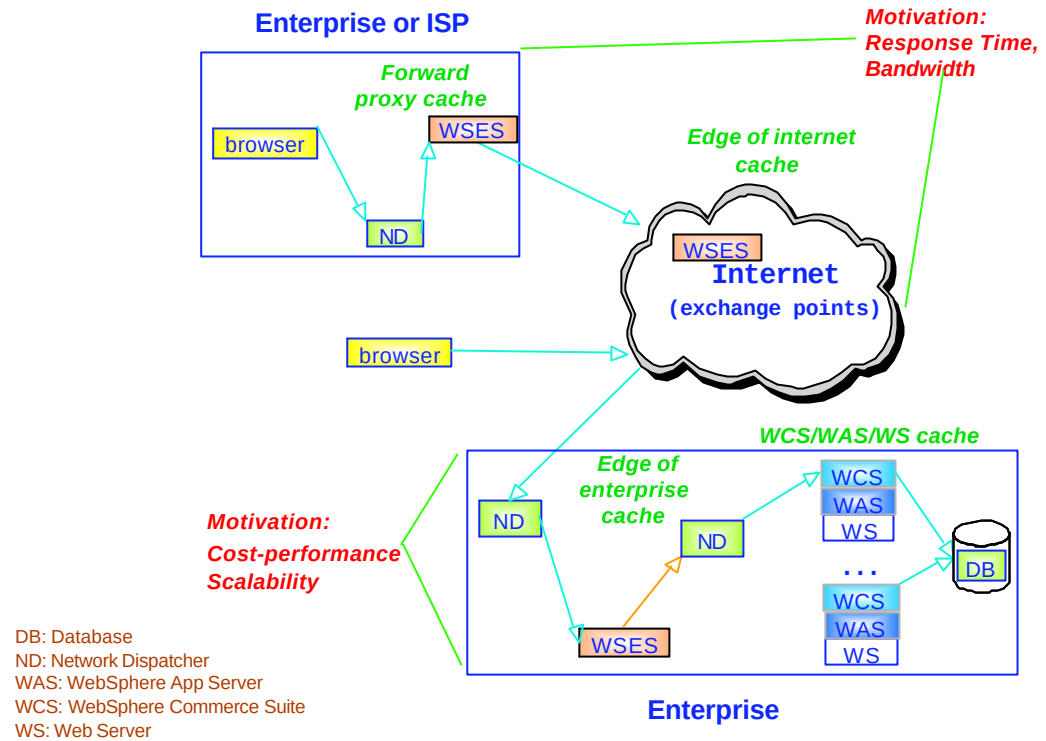
What, Where, When and How

Considerations for caching in web context are:

- What, when and where to cache
- Granularity of caching: web pages, fragments of pages, servlet execution result, SQL query result, data, ...
- Location of cache: client, proxy, edge-of-net, internet service provider (ISP), edge-of-enterprise, app server, web server, DBMS
- Caching and invalidation policies: application transparency, push vs. pull, freshness maintenance, triggers, log sniffing
- Enabling cache exploitation: routing, failover, accounting, authentication, authorization, ...
- Tools: performance monitoring, analysis

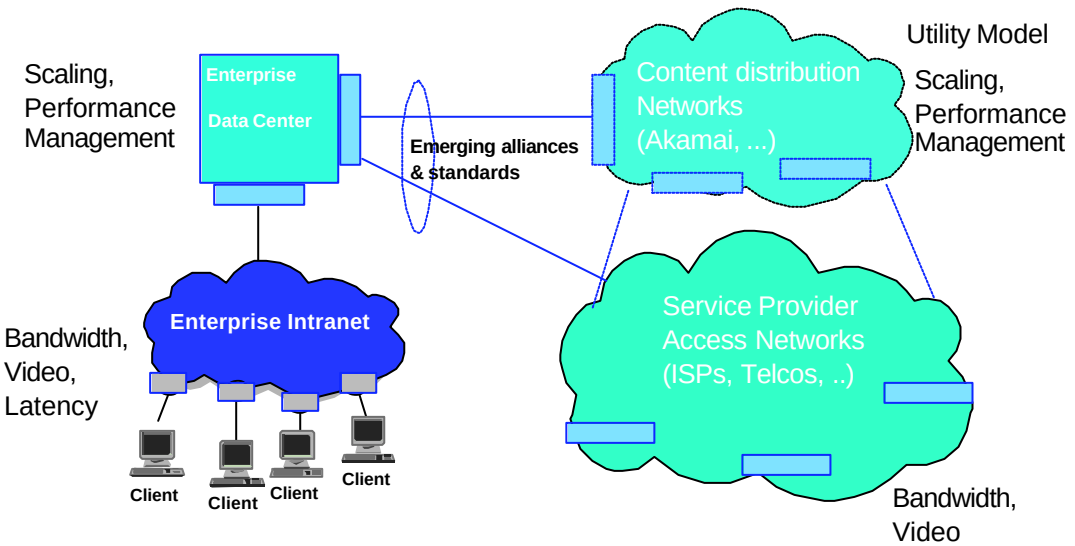
Related DB Technologies: replication, materialized views, mediator systems, client-server DBMSs, buffer management, main-memory DBMSs, query optimization, content mgmt, ...

Common Points of Caching (Non-Database)



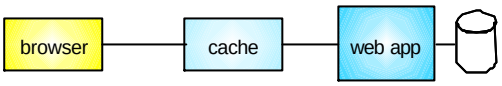
DB: Database
ND: Network Dispatcher
WAS: WebSphere App Server
WCS: WebSphere Commerce Suite
WS: Web Server
WSES: WebSphere Edge Server

Edge Serving



Cache Models

Front-End Cache

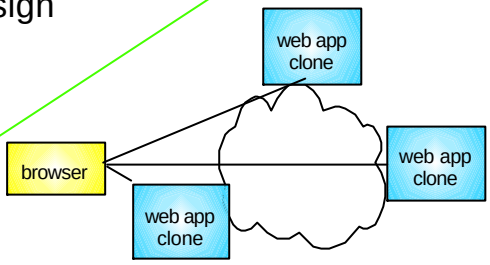


- caches data + markup
- cache can be app independent
- static pages easily cached
- dynamic pages need app support

Data Cache

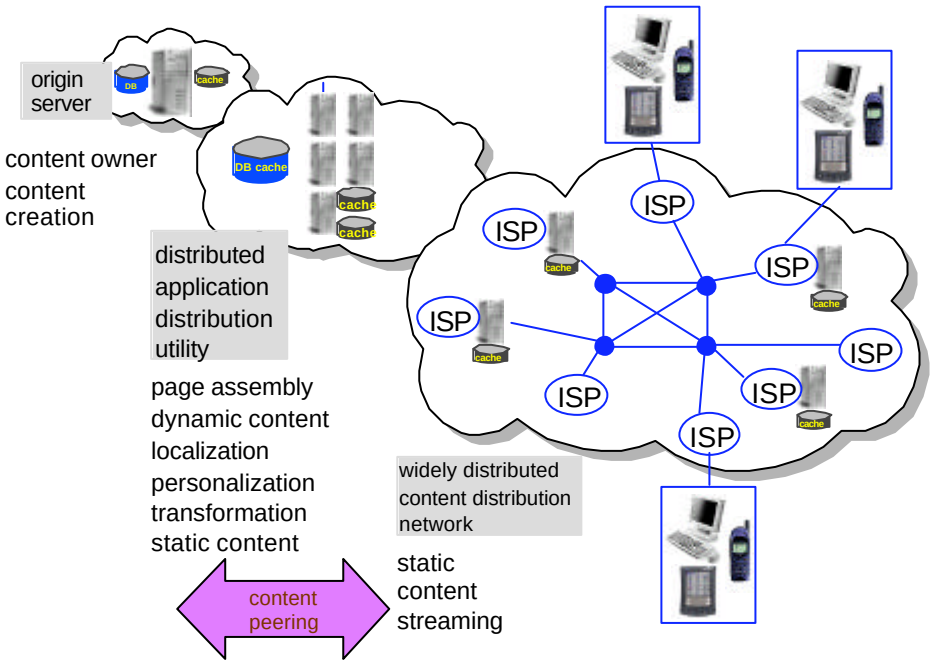
- caches data
- effectiveness depends on app design

Distributed App



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

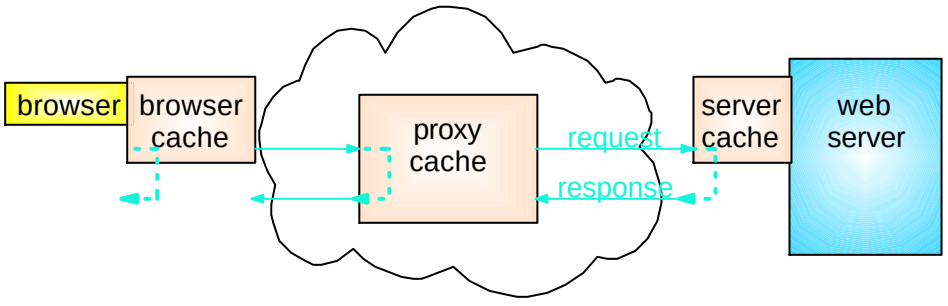
Tiers of Content Serving



Specialized Caching Vendors/Software

- Content Delivery Network (CDN) 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 net cache SW: IBM WebSphere Edge Server

HTTP Caching Today



- Multiple caches - 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

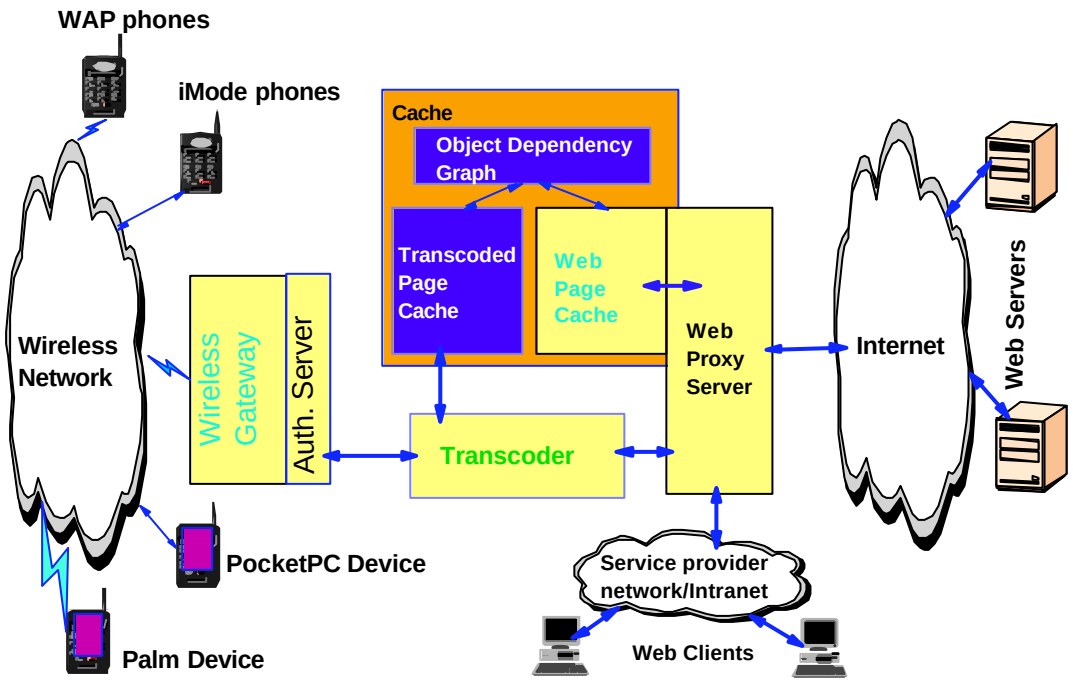
Dynamically-Generated Pages

Increased generation due to

- Database-centric e-commerce apps
- Frequently updated content
- Personalization
- Access device differences

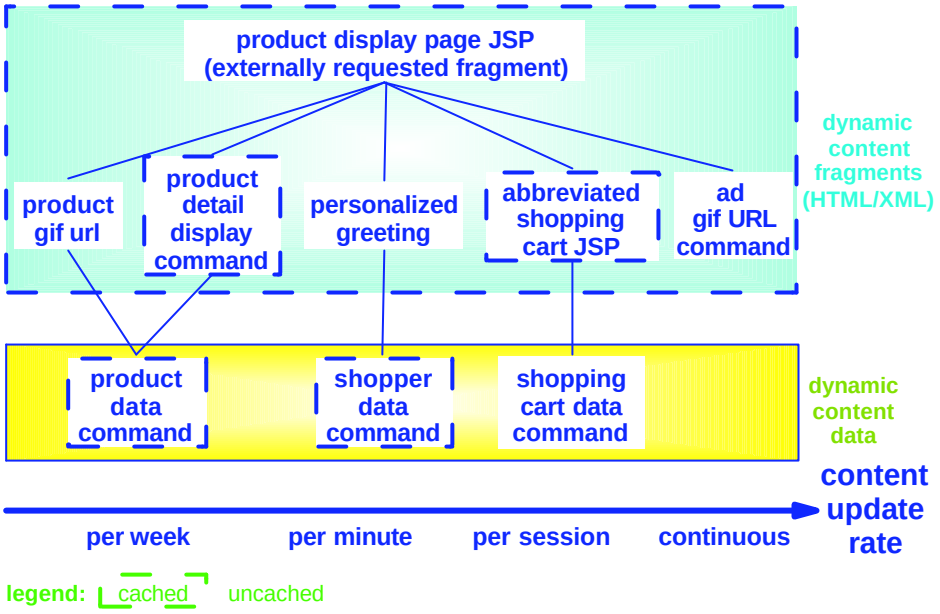
Proxy caching is ineffective for such pages

Multi-Device and Mobile Web Caching



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

EJBs and Caching

Container can select from 3 commit-time options:

- Option A: Container caches “ready” instance between transactions. Container ensures instance has exclusive access object state in persistent store.Therefore, Container doesn't have to synchronize instance’s state from persistent store at start of next transaction.
- Option B: Container caches “ready” instance between transactions. Container doesn't ensure that instance has exclusive access object state in persistent store. Therefore, Container must synchronize instance’s state from persistent store at start of next transaction.
- Option C: Container doesn't cache a “ready” instance between transactions. Container returns instance to pool of available instances after a transaction completes.

EJBs and Caching

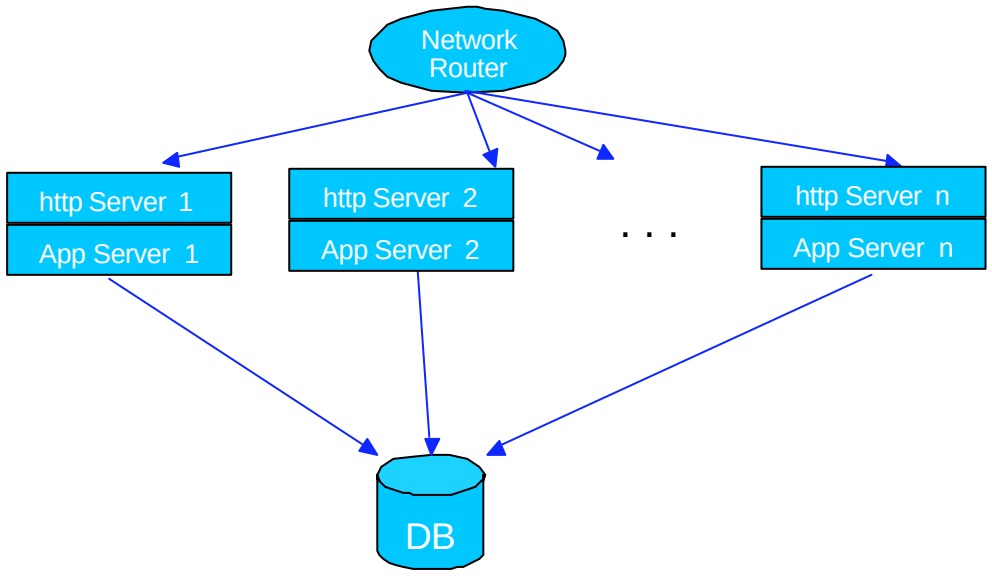
Option A caching incompatible with clusters and shared data

- When Option A caching is in use, app server hosting enterprise bean container must be the only updater of data in persistent store. Hence, Option A is incompatible with:
  - Workload managed servers (such as a cluster of clones)
  - Database with data being shared among multiple apps
- Shared database access corresponds to Option C caching

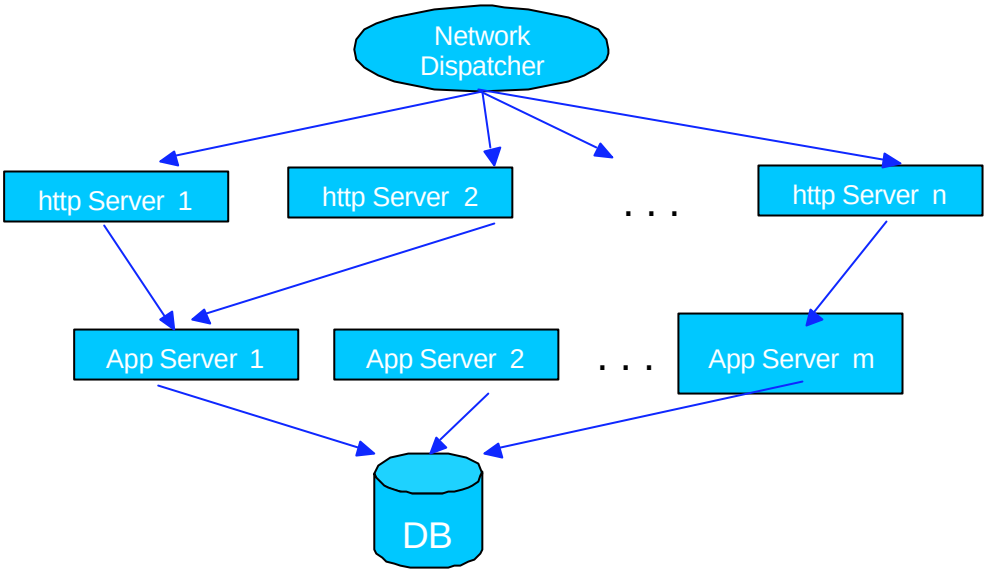
Database Caching

- 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 thruput, congestion control, availability, QoS

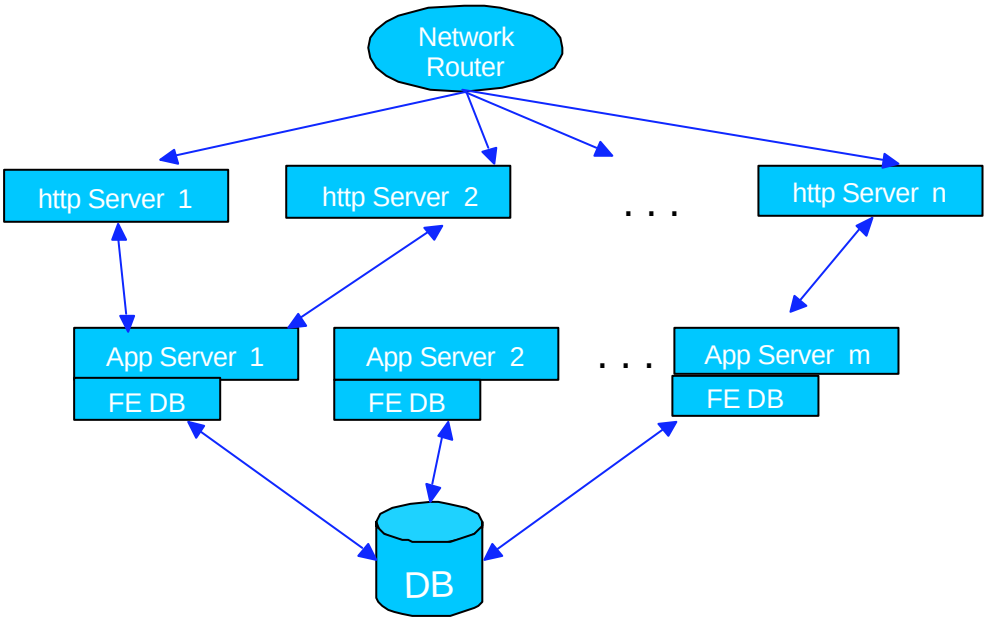
Classical Web Setup - 3 tier



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 (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 shouldn't have to change
- App's DB schema shouldn't have to change
- Support failover of nodes
- Support reasonable update semantics
- Support dynamic addition/deletion of app server nodes
- Limits on update propagation latencies

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 DBs in cache nodes
- Cons
  - Complex routing: integrate with edge server
  - Complexity in DDL spec and query processing
  - Complex update logic - 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'))
- Query rewrite routes query to materialized view instead of base table
- Refresh: Incremental/Deferred
  - 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 invalidates cached content
- Mutual consistency of related data (transaction guarantee)

Updates - Push Vs Pull

Push Pull

- Push advantages
  - reduced response time for first hit
  - overwrite: less total path length than invalidation + pull
- Pull advantages
  - everything doesn't have to fit in cache
    - not in cache until accessed
    - only hottest stay in cache long term
  - personalized info cached where needed, not everywhere
  - easy to get context required to execute JSP - a real request will be underway when executing a JSP

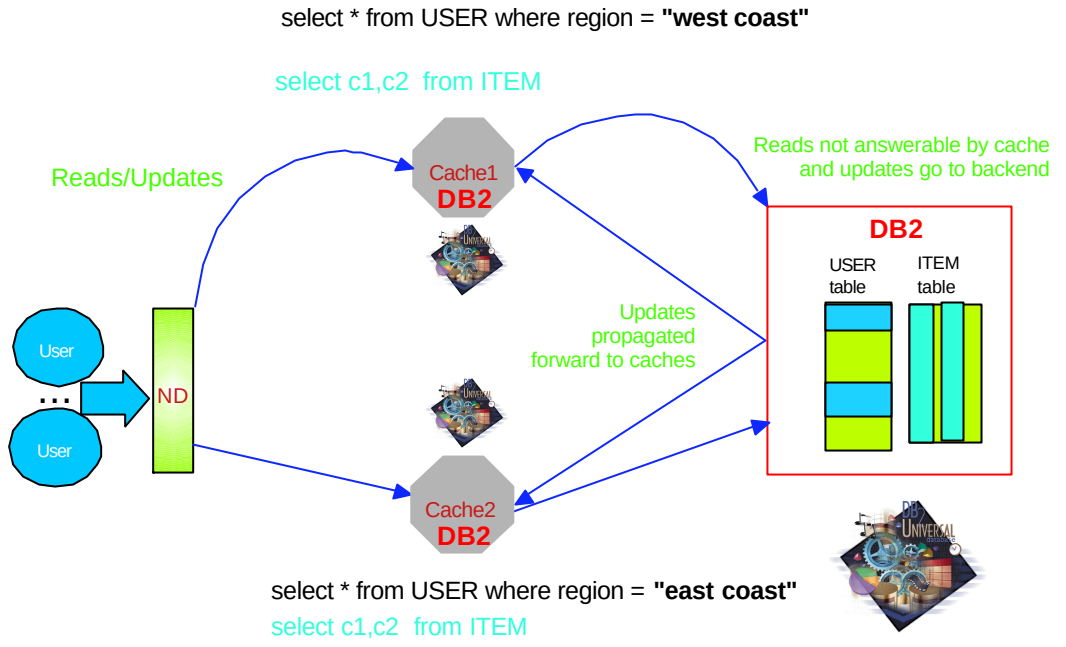
Update Handling

- Where to perform the updates first
  - Frontend only
  - Backend only
  - Both places (2-phase commit - cost, availability issues)
- Asynch propagation if update performed in only one place
- Semantic problems
  - Users not seeing their own updates
  - Update replay on other copies (e.g., on logical redo if identity columns are involved)

Data Caching Choices

- If cache in DB process, app server will incur costs of
  - process boundary crossing
  - data conversions
- Alternative: cache data in app server process in
  - relational form (JDBC query results) for servlets, session/entity beans
  - Java object form 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 support
- DBMS is told front-end data is subset of backend data
  - Conditionals 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
- Data freshness guarantees and query options

Challenges

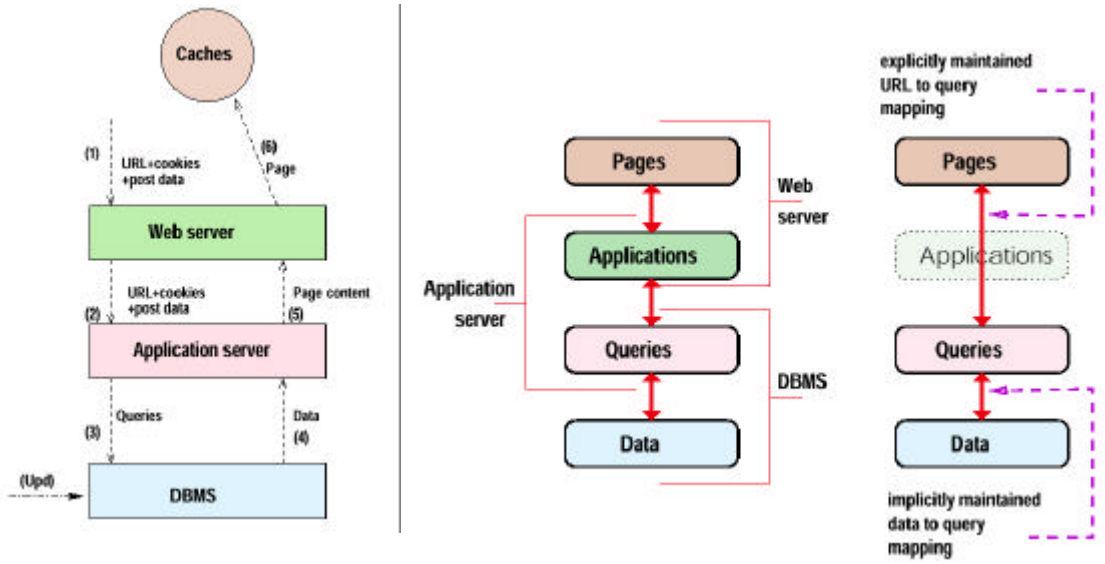
- Describing cache contents
  - Expand on capability of materialized views
- Replication subscriptions: How to quickly do dynamic modifications?
- Intelligent routing by WebSphere Edge Server
- Query workload analysis to determine view definitions
- Tracking changing workload
- Handling of updates & associated read semantics
- Query optimization
- Failure handling
- DBA tools

Query Result Caching

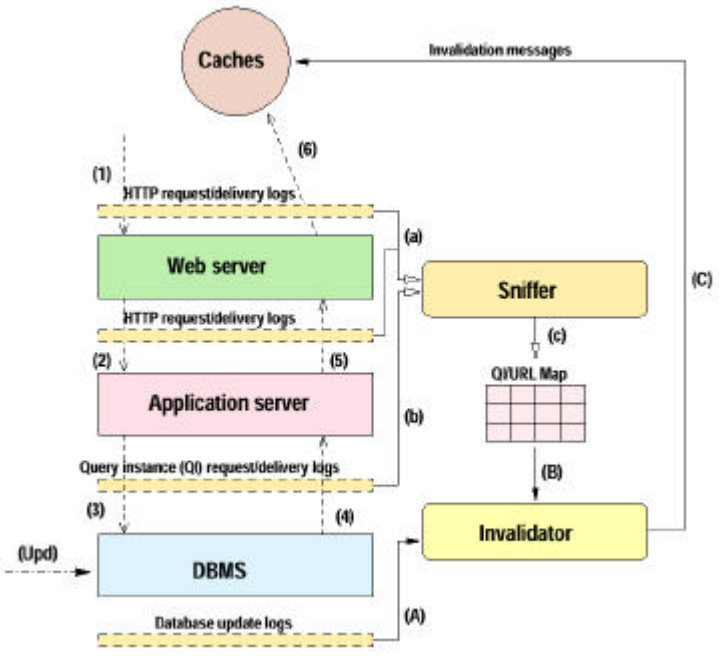
- Cache could exist within or outside DBMS (e.g., integrate with JDBC driver)
- Results Tracking
  - Individual results kept separately
  - Results combined to avoid duplicate subsets
- Cache used to answer
  - only repeated (exact match) queries: just return bits in bucket, no need for complex query engine
  - any query for which answer is in cache (subset or union of earlier queries): need query processing capabilities
- Full exploitation of external-to-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's CachePortal

- Automation of extracting URL-query mapping (sniffer)
- Log-based invalidator



NEC's CachePortal

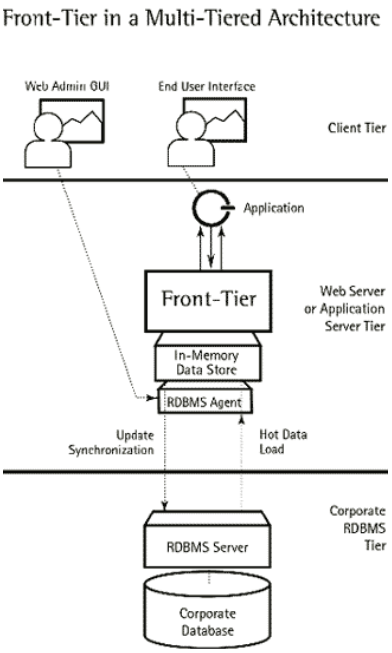


Database Caching Products

- 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 persistent cache and edge of net
- Main memory technologies being exploited in some cases

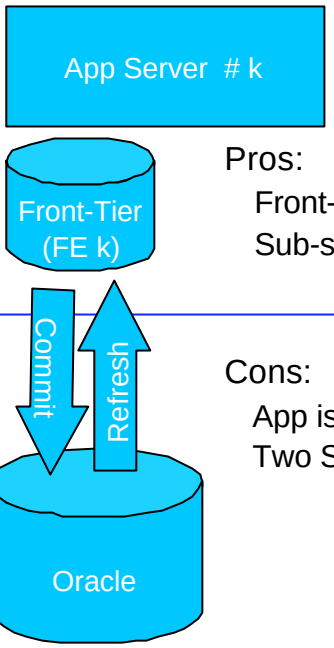
TimesTen

- Original HP Lab main-memory project spun out as startup
- MMDB product TimesTen
- Front-Tier: Cache product for Oracle
- 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/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



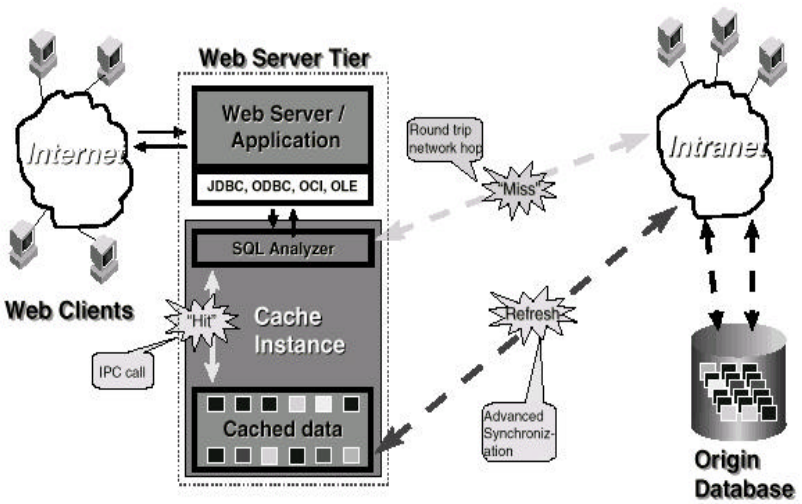
Pros:

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

Cons:

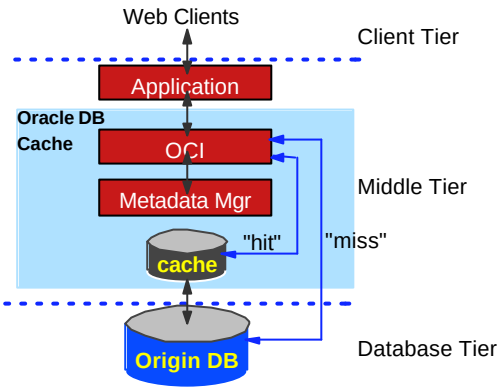
- App is aware of Front-Tier
- Two SQL dialects

Oracle 9i IAS - Relational Cache



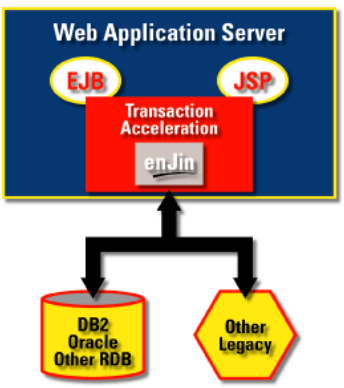
Oracle 9i IAS - Relational Cache

- Cache only entire tables and not subsets of table contents
- Queries processed entirely using cache or entirely sent to backend; updates always sent to backend; only one backend DBMS per app server
- Tooling for set up and monitoring cache performance
- Use of cache: allows specification on a per SQL statement basis, DB connection level or globally
- 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



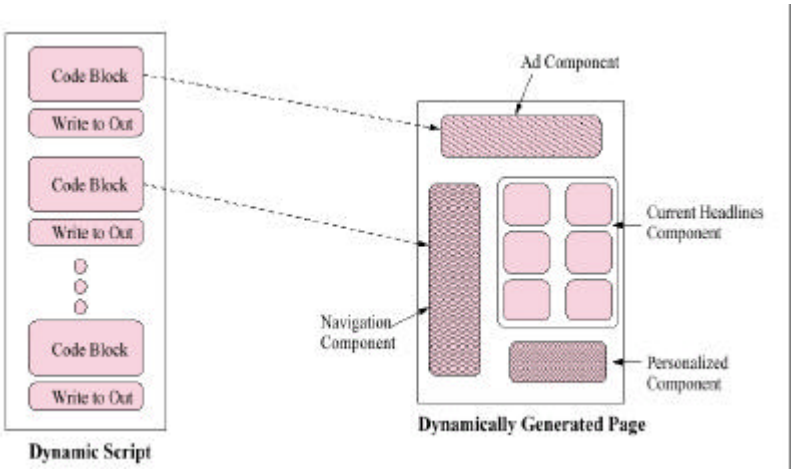
Versant's enJin

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



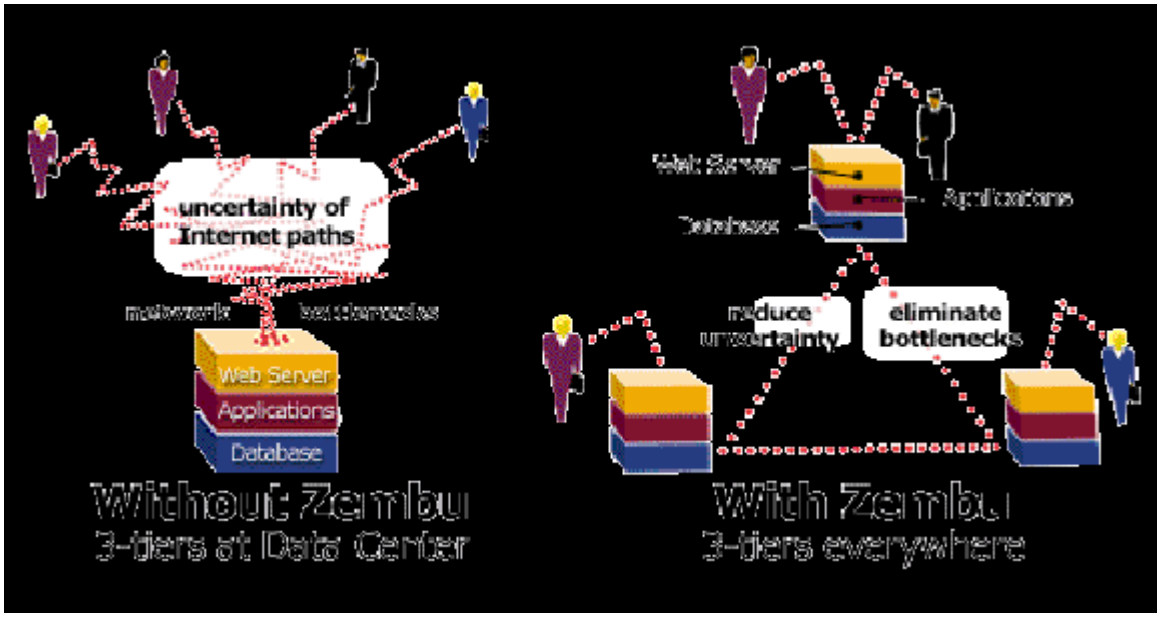
Chutney Tech's Dynamic Content Accelerator

- Fragment-level caching of HTML
- Observation and prediction based techniques for cache replacement and invalidation



Zembu

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



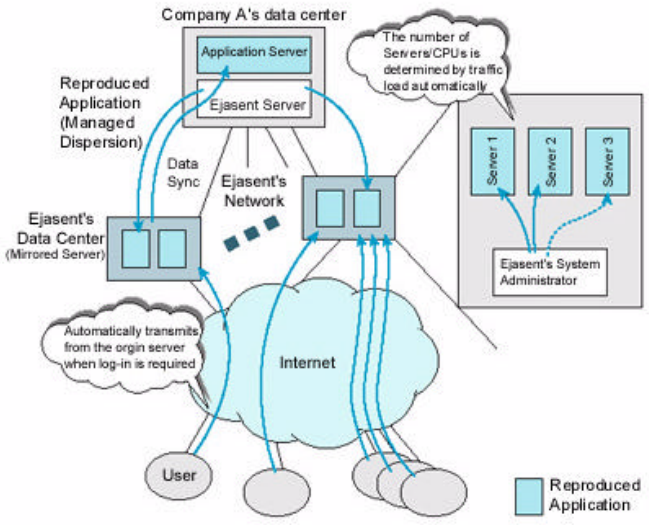
Zembu

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

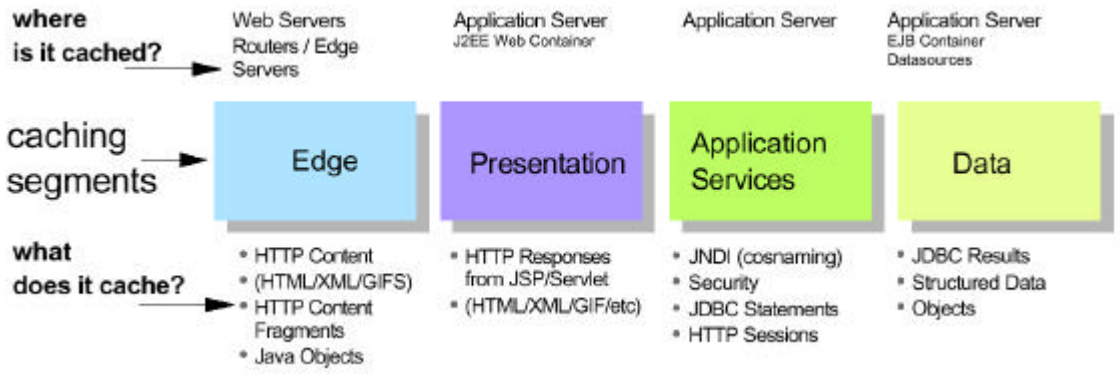
- Startup owns and operates a distributed, interactive service net
- Users in real time allocate additional resources
- BEA and Broadvision Solaris apps - snapshot of binary images captured and run on Ejasent servers



Case Studies

- IBM WebSphere
- Olympics
- eBay

Potential WebSphere Caching Landscape

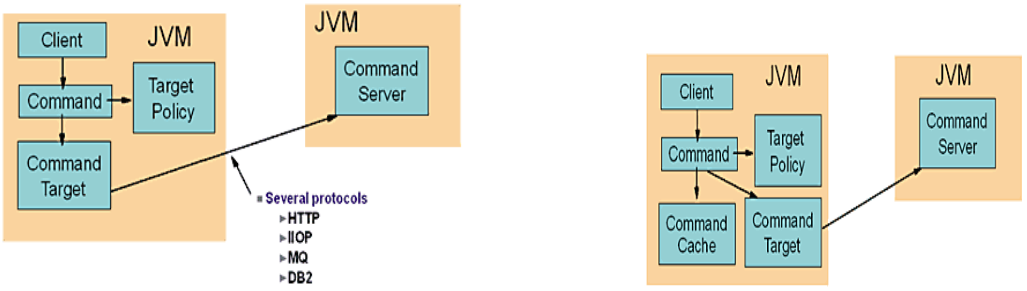


Dynamic Caching (WebSphere V4.0)

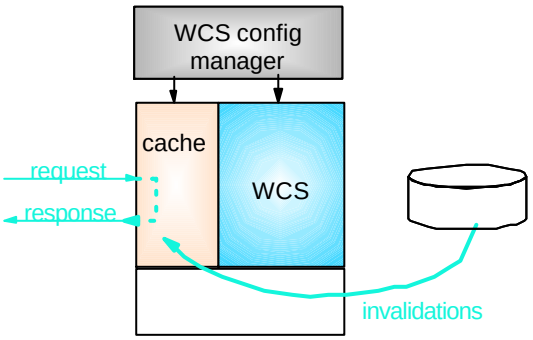
- Caches Servlet/JSP response, JNDI, Security, Statements, HTTP Session
- Application developers/assemblers control how fragments are cached using XML cache descriptor
- Define rules based on servlet, URI, request attribute/parameter/session/cookie
- Rule/time-based, and programmatic techniques for invalidating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

WebSphere Commands and Caching

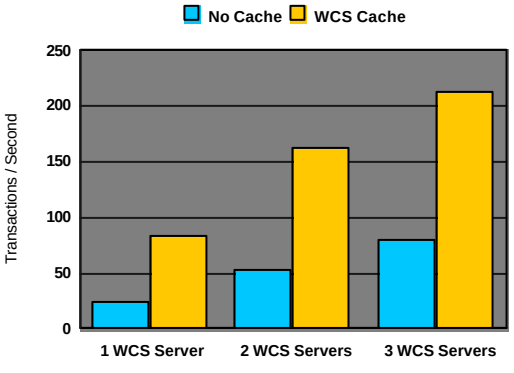
- Command: JavaBean that represents a particular operation; can be serialized
- Commands isolate apps using them from the implementation of the services they reach through them
- Context needed to execute a command is carried in the command via setable properties
- Results of executing a command are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



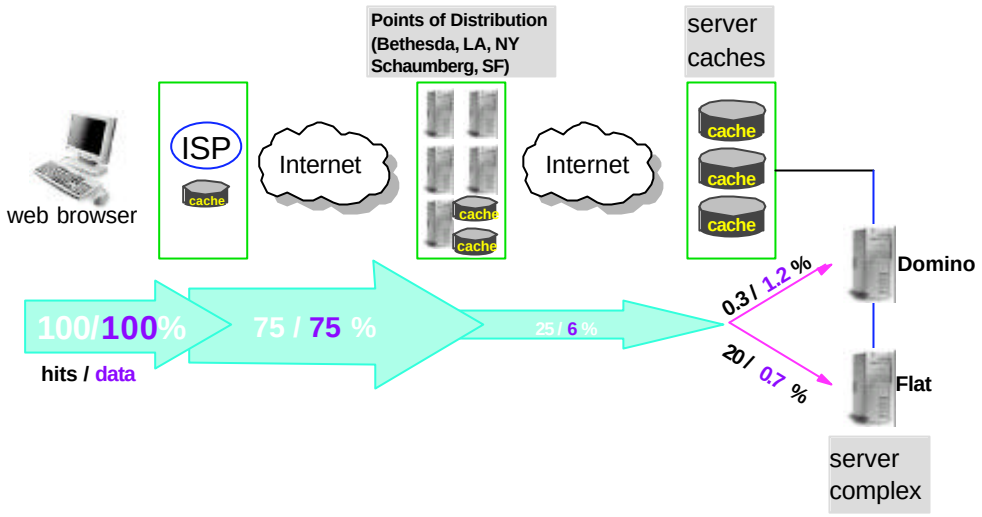
WebSphere Commerce Suite (WCS) Cache



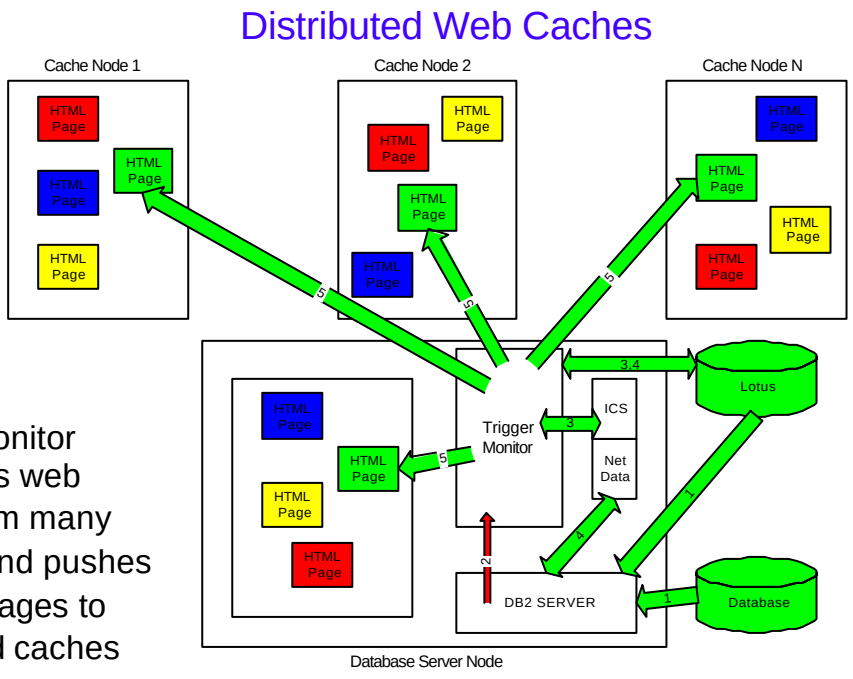
- Usually for caching catalog data
- Admin influences what gets cached
- Now, personalization turns off caching



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 or so
- App level routing of queries to FE or BE

As of October 2000

- 18.9M registered users
- Items in 4,500 categories
- Millions of items for sale
- Users added 600K new items daily
- 4M page views per month
- Daily reach topped 16.2%, # of unique visitors on an average daily basis was 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
- Cache content purging algorithms
- Performance monitoring and tuning DBA tools
- Load balancing or cache-intelligent URL routing
- Describing cache contents for use in query rewrite
- 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

Sample of Recent Research Work

- VLDB2001: Labridinis, Roussopoulos
  - With continuous updates, scheduling refresh of cached views to maximize QoD (Quality of Data) based on cost of updating and popularity of views - Quote.com study
- VLDB2000: Yagoub, Florescu, Issarny, Valduriez
  - Relies on declarative spec of web sites; caches relations, XML fragments and html files
- VLDB2001: Luo, Naughton
  - Proxy caching of form-based queries (*active* caching); a servlet implements some DBMS functionality in proxy and user provides mapping of forms to query templates

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", L. Degenaro, A. Iyengar, I. Lipkind, I. Rouvellou. Proc. ACM/IFIP Middleware 2000, Palisades, April 2000, Springer-Verlag.
- "High-Performance Web Site Design Techniques", A. Iyengar, J. Challenger, D. Dias, P. Dantzig. IEEE Internet Computing, March/April 2000.
- "A Publishing System for Efficiently Creating Dynamic Web Data", J. Challenger, A. Iyengar, K. Witting, C. Ferstat, P. Reed. Proc. IEEE INFOCOM, March 2000.

References

- Brian D. Davison's Web Caching and Content Delivery Resources, http://www.web-caching.com/
- "Scaling Up e-Business Applications with Caching", M. Conner, G. Copeland, G. Flurry. DeveloperToolbox Magazine, August 2000. http://service2.boulder.ibm.com/devtools/news0800/art7.htm
- "Enabling Dynamic Content Caching for Database-Driven Web Sites", K.S. Candan, W.-S. Li, Q. Luo, W.-P. Hsiung, D. Agrawal. Proc. SIGMOD, Santa Barbara, May 2001.
- "A Comparative Study of Alternative Middle Tier Caching Solutions to Support Dynamic Web Content Acceleration", A. Datta, K. Dutta, H. Thomas, D. VanderMeer, K. Ramamritham, D. Fishman. Proc. VLDB, Rome, September 2001.
- "Form-Based Proxy Caching for Database-Backed Web Sites", Q. Luo, J. Naughton. Proc. VLDB, Rome, September 2001.
- "Update Propagation Strategies for Improving the Quality of Data on the Web", A. Labrinidis, N. Roussopoulos. Proc. VLDB, Rome, September 2001.
- "Cache Portal: Technology for Accelerating Database-Driven e-Commerce Web Sites", W.-S. Li, K.S. Candan, W.-P. Hsiung, O. Po, D. Agrawal, Q. Luo, W.-K. Huang, Y. Akça, C. Yilmaz. Proc. VLDB, Rome, September 2001.
- "Versant enJin for IBM WebSphere", http://www.versant.com/products/enjin/whitepapers/WP_9908-rev0103r4W.pdf

References

- "Adaptive Push-Pull: Dissemination of Dynamic Web Data", P. Deolasee, A. Katkar, A. Panchbudhe, K. Ramamritham, P. Shenoy. Proc. WWW10, Hong Kong, May 2001.
- "Engineering server-driven consistency for large scale dynamic web services", J. Yin, L. Alvisi, M. Dahlin, A. Iyengar. Proc. WWW10, Hong Kong, May 2001.
- "Adaptive Leases: A Strong Consistency Mechanism for the World Wide Web", V. Duvvuri, P. Shenoy, R. Tewari. Proc. IEEE INFOCOM, Tel Aviv, March 2000.
- "Design Considerations for Distributed Caching on the Internet", R. Tewari, M. Dahlin, H.M. Vin, J.S. Kay. Proc. ICDCS, Austin, June 1999.
- "Caching Strategies for Data-Intensive Web Sites", K. Yagoub, D. Florescu, P. Valduriez, V. Issarny. Proc. VLDB, Cairo, September 2000.