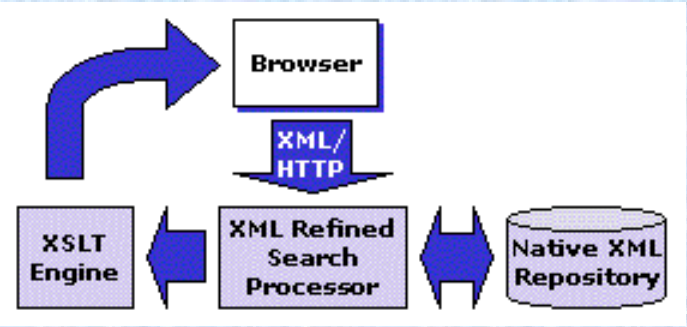
XML Internet Search Repository

Addressing e-Business problems that fall between full-text search engines and standard databases

XDisect
XML Index & Intelligent Search

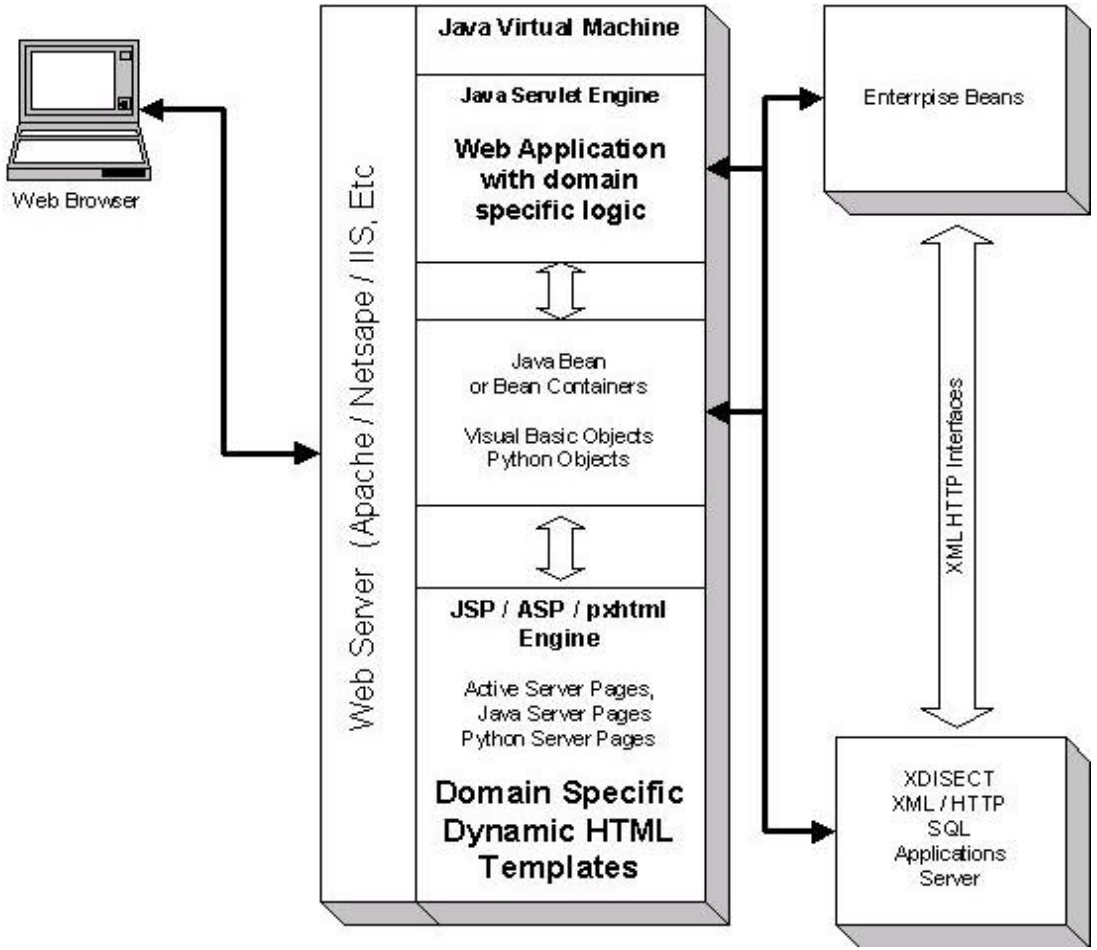


XML search infrastructure

- **Enable rapidly evolving & collaboration centric e-businesses to manage the impact of diversity and frequent data change on IT infrastructure.**
- **Augment RDBMS and LDAP engines by adding full text search to any attribute.**
- **Enable the use of highly variable data structures**
- **Enable rapid evolution of data model.**
- **Deliver more relevant results using the structure of the data.**
- **Reflect Incremental document changes immediately.**

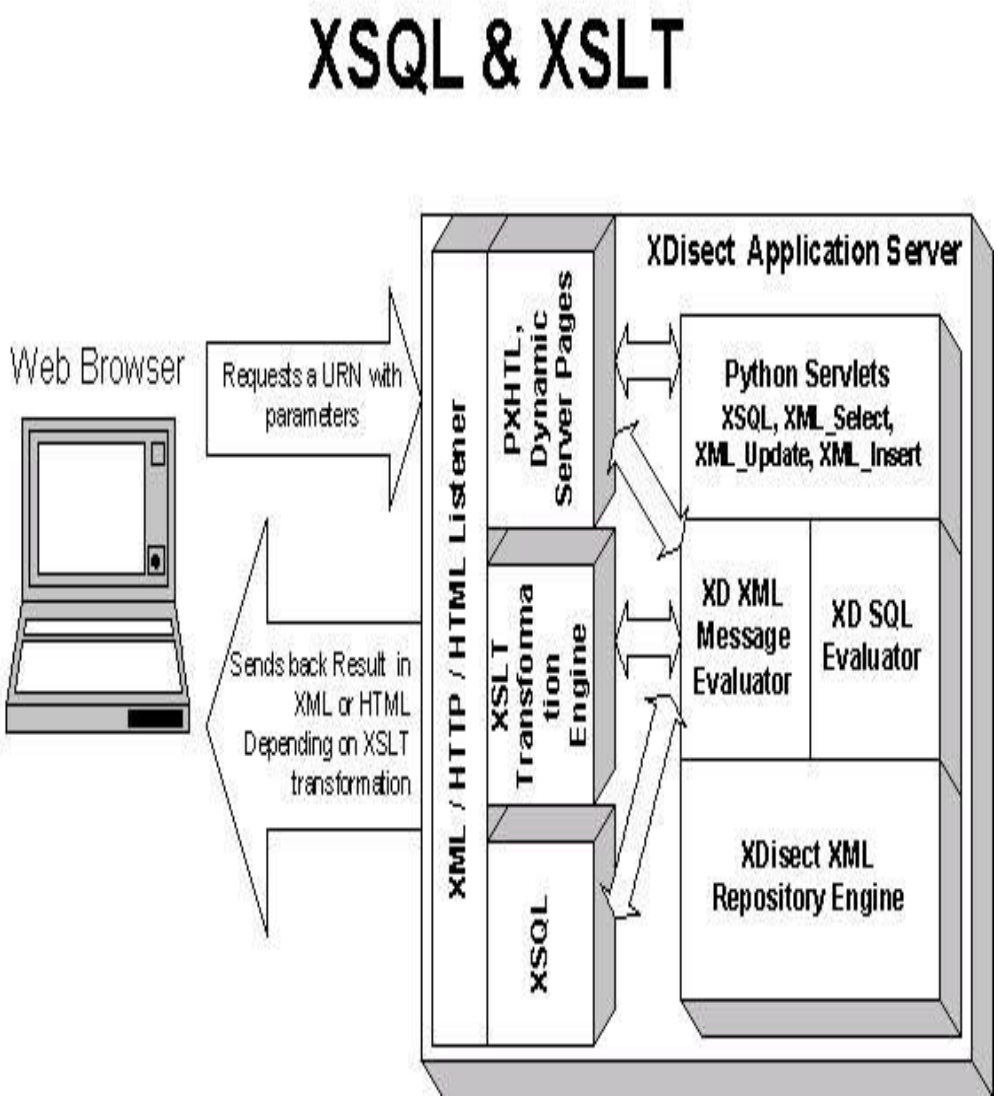
JSP/J2EE Architectural View

XDisect can be accessed from JSP & EJ beans



- XDisect can be accessed from any J2EE platform
- Open XML/HTTP protocol
- The front-end JSP & beans can also access XDisect.
- The app server view of this is more complicated but most JSP&EJB components will flawlessly move into the app server platform.
- Some advanced app server capabilities are not adequately utilized in this simplified view.

XSL Architectural View



- XSQL – Server side stored XML queries. A standard published by Oracle.
- Use XSQL to query XDisect.
- Use XSLT to transform query results into different XML or html formats.
- On the front-end an application server like BlueStone or BEA-WebLogic is used rather than direct connection to the browser.
- For security a reverse proxy between public internet & XDisect application server is recommended.

Interaction with Server – Insert, Update, Delete XML

Open XML /HTTP Interfaces

Sample Insert

```
<xml_insert><record>
    <person>
      <user_id>gkill2</user_id>
      <soc_sec_no>555-555-5555</soc_sec_no>
      <name>
          <first>gill</first><last>killroy</last>
      </name>
     <email>gill@killroy.com</email>
    <phone> <office>(408)364-1741</office></phone>
    <interest>programming</interest>
    </person>
</record></xml_insert>
```

Sample Update

```
<xml_update>  <update>  <cmd>
    UPDATE  WHERE  person*name.first  EQ "gill"
    AND person*last   CONTAINS "kill“
    SET person.name.first = “Licra”,
          person.phone.cell  = “408-343 6347”
 </cmd> </update> </xml_update>
```

Sample Delete

```
<xml_delete><delete>
     DELETE WHERE  person*name.first EQ “gill”
        AND person*last  CONTAINS “kill”
 </delete> </xml_delete>
```

- Supports insert, update & delete.
- Records can be batched.
- XML records do not need a pre-defined schema
- Schemas can be used for validation, but not mandatory.
- Update & Delete commands use a simple SQL-style syntax.
- Sub-second response time for most insert, update & delete commands.
- Updated data is available for querying instantaneously
- Updates support full merge, replace or additive semantics.

Interaction with Server – Queries
Open XML /HTTP Interfaces

Server Side Stored Queries

```
<?xml version="1.0"?>
<xsql:query connection="demo"
    xmlns:xsql="urn:pybiz-xsql">
        SELECT *  FROM person
          WHERE person.user_id EQ "{@id}";
</xsql:query>
```

Fuzzy Path Spec Queries

```
SELECT  person*name, person*phone
WHERE person*skill* CONTAINS “java”;
```

Direct Queries

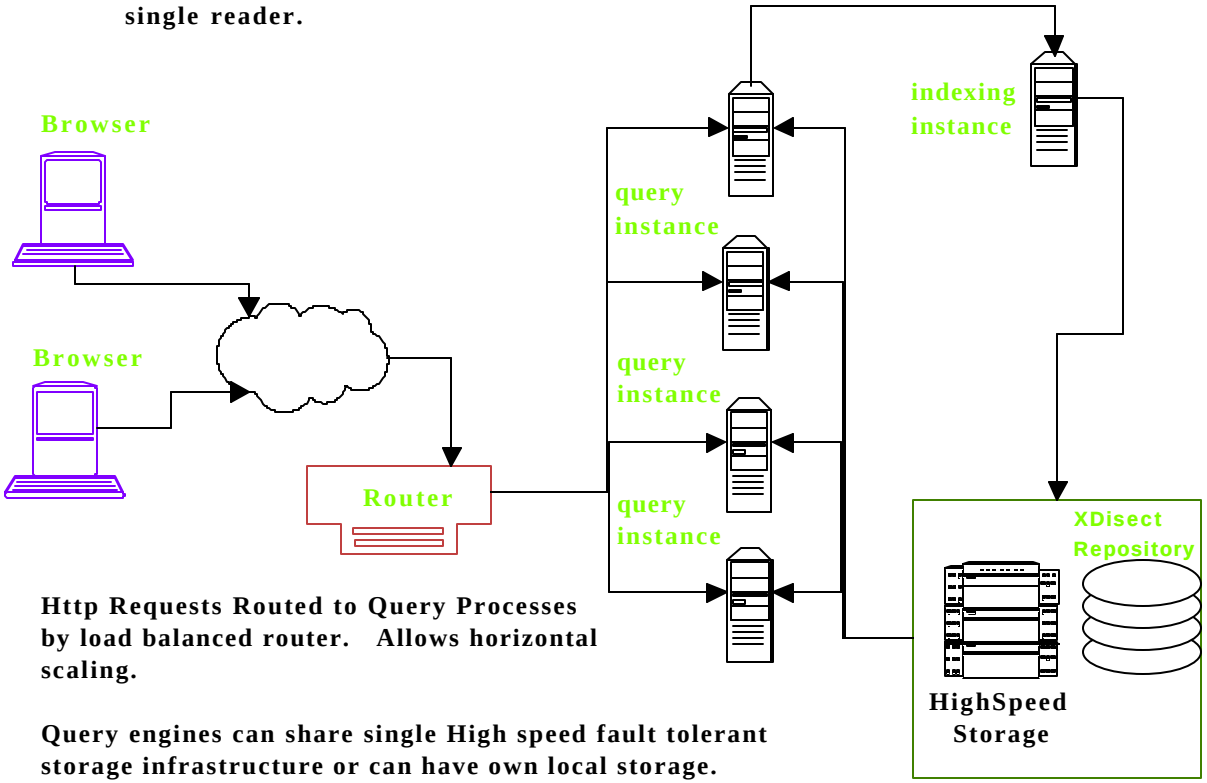
```
<xml_select>
  SELECT  WHERE
     person.interest CONTAINS
"programming";
</xml_select>
```

- server-side & ad-hoc queries.
- Server-side queries allow cgi-style passing of values for variables.
- Query language syntax based on SQL.
- Xpath-style syntax for querying nested data.
- Sub-second query response time
- Supports numeric comparison
- Supports soundex searches on any keyword in any tag.
- Single query can return different shapes and types of XML records.
- Queries can return full XML or specified parts of the XML record.
- Supports XSLT to transform query results into different XML or HTML formats

Scalability

Horizontal scaling using multiple reader agents - Shared Storage

Any query instance can receive update commands but they are all processed by single reader.



Http Requests Routed to Query Processes by load balanced router. Allows horizontal scaling.

Query engines can share single High speed fault tolerant storage infrastructure or can have own local storage.

Many reader single writer design makes query scaling easy.

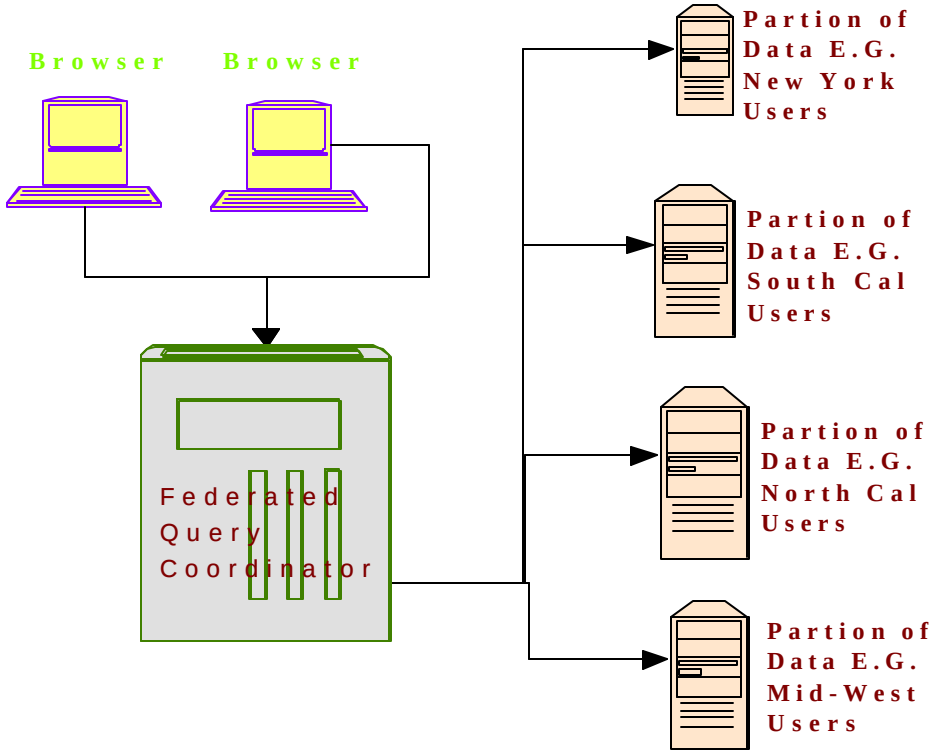
Support for synchronizing multiple data stores when query machines are using local storage.

Designing Data Partitions for Scale

Federated query management

Federated Query Partitions Data for scaling optimal Query Response

- Each partition can store different data.
- Each partition internally may have load balanced QOS infrastructure.
- Coordinator aggregates results from different partitions.
- Coordinator manages paging for clients.
- Partitions do not need to be XDisect Based.



Addition of a Load balancing router can add fail over and horizonal scaling to the query instances

Intermediate applicaton servers such as Bluestone not shown for simplicity.

Performance metrics

Easily optimized using horizontal scaling techniques.

Query Response Time

- Basic Queries : 0.036 secs
- Avg. response : 0.1 secs
- Complex queries : 0.2 secs

Insert, Update Speed

- Insert – 3MB per min.
- Delete & Re-Insert – 2Mb per min.

Database Size

- Tested with DB > 2GB
- Almost linear query perf. as DB grows increases

Query Performance Factors

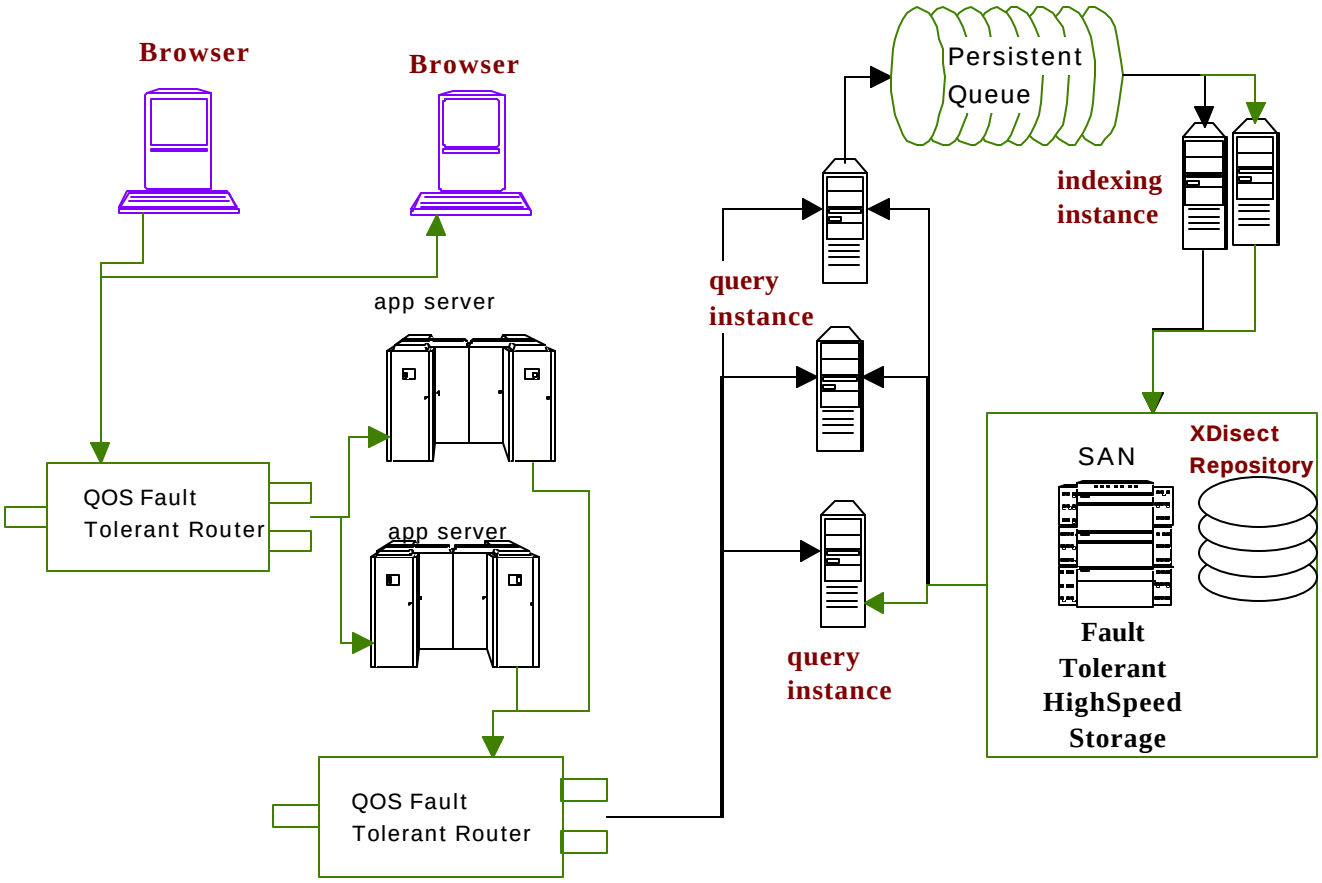
- **# of Concurrent Readers**
- **Avg. size of result record, # of records returned.**
- **Size of original records.**
- **Query complexity**
- **# of ambiguous terms to resolve.**
- **concurrent insert/update operations.**
- **Total repository size.**
- **Ability to optimize Query.**

Test Environment for above results

- **900 Mhz Athalon, single CPU**
- **512 MB Ram, 60GB IDE Drive**
- **500 MB DB, avg record size – 4K**
- **XML nested upto 4 levels deep**

Fault Tolerance

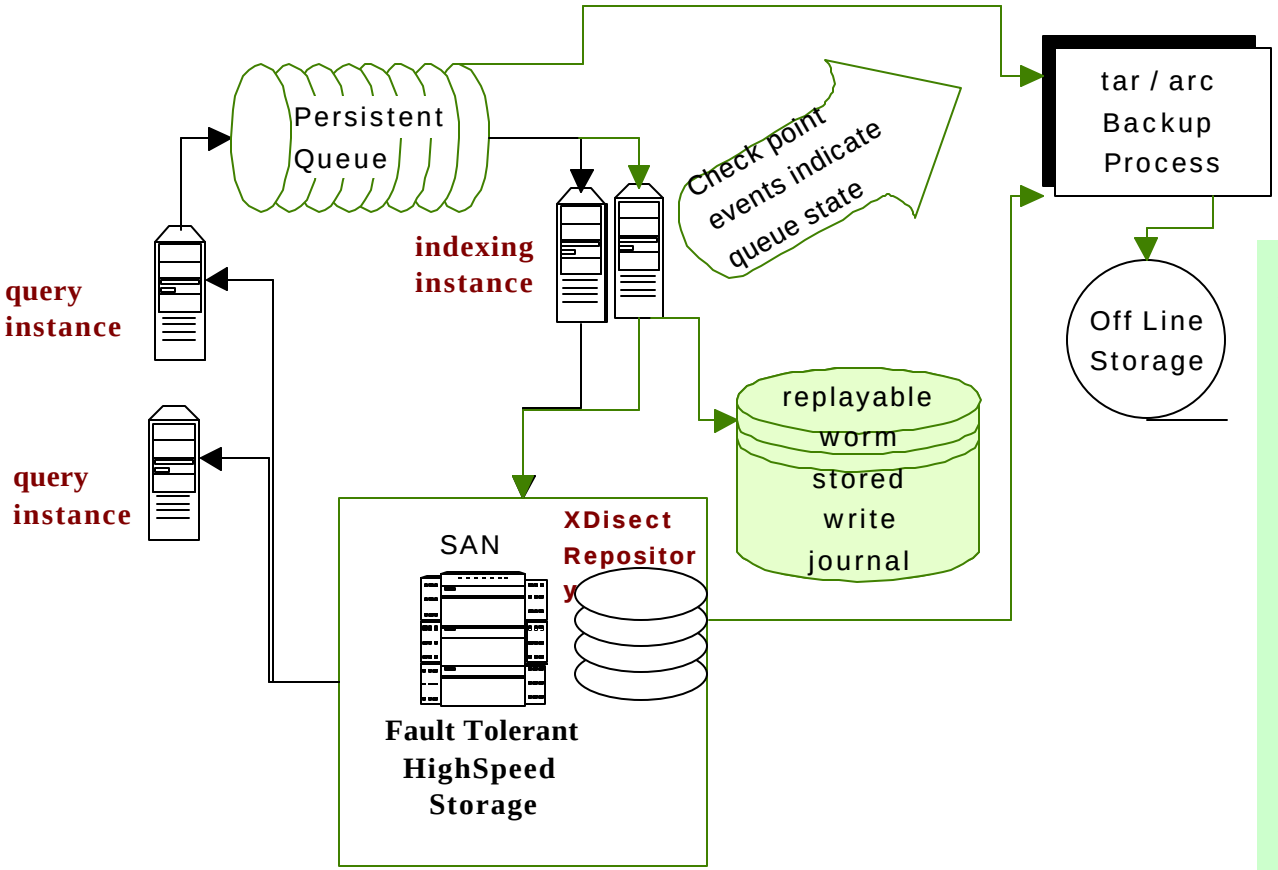
utilizes standard QOS Routers



- Any query instance can receive requests for insert/update
- Requests are stored in a persistently queue.
- No single point of failure in the architecture.
- Total failure would require failure of all components
- Second router functionality can be done using software but we prefer the h/w router for performance.
- Only one instance of the XDisect Indexer is active at a time.

Hot Backup & Recovery

backup any time, ASCI write journalling



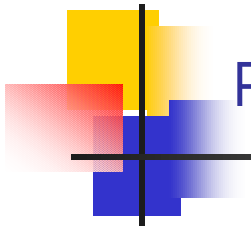
- Safe to backup at any time.
- Recovery involves
  - **shut down**
  - **restore from tape**
  - **Replay journal into command queue for restoring since backup.**
  - **Restart processes.**
  - **Wait for command queue to empty**
- Backup leverages standard IT backup tools & processes.

Transactions & Rollback

All or Nothing semantics

```
<xml_update>  <update>
   <cmd>
     UPDATE REPLACE WHERE contact.who EQ
      “gkill"  AND contact.id EQ "999981“;
   </cmd>
   <record><contact>
       <id>999981</id>
       <date> Jan 13, 1979</date>
       <who>gkill2</who>
       <reason>sales follow through</reason>
       <result>need 500 more people</result>
    </contact></record></update>
 <update><cmd>
     UPDATE
     DROP contact.next_contact
     SET contact.skill = ['java', 'cobol']
     WHERE contact.id EQ "999983"
   </cmd>
 </update>
</xml_update>
```

- Submit multiple requests in same batch.
- Specify that all request need to be committed successfully (Optional).
- Rollback on failure.
- No support for nested transactions
- No support for two phase commit



Problem focus for Native XML & Relational Engines

Native XML

- High speed search
- Search on any word in any element.
- Soundex on any word in any element.
- Semi-structured search.
- Enables schema evolution.
- Store & retrieve different XML structures for same class of data.

Relational

- Transactional
  - Rollback
  - Locking
- Acid Properties
- High speed inserts
- High speed search when pre-defined indexes.
- Snapshot
- Proven trust relationship with IT

Applications focus for Native XML & Relational Engines

Native XML

- Partially understood problems.
- Rapidly evolving business.
- Strong partnering & collaboration needs.
- Data model is constantly changing.
- Need rapid time to market

Examples

- Secure collaborative Portals
- CRM aggregation
- Supply Chain Management SCM
- Wireless service aggregators
- On line exchanges

Relational

- Well constrained problems
- Well understood problems
- Well defined schema
- Controlled rate of change.
- Stable business environment.

Examples

- Financial applications
- Inventory control
- Billing systems

XDisect Enables B2B Collaboration