



BISSA: Empowering Web gadget Communication with Tuple Spaces

Udayanga Wickramasinghe, Charith Wickramarachchi,
Pradeep Fernando, Dulanjanie Sumanasena, Sanjiva
Weerawarana, Srinath Perera
Lanka Software Foundation, WSO2 Inc., University of
Moratuwa



Outline

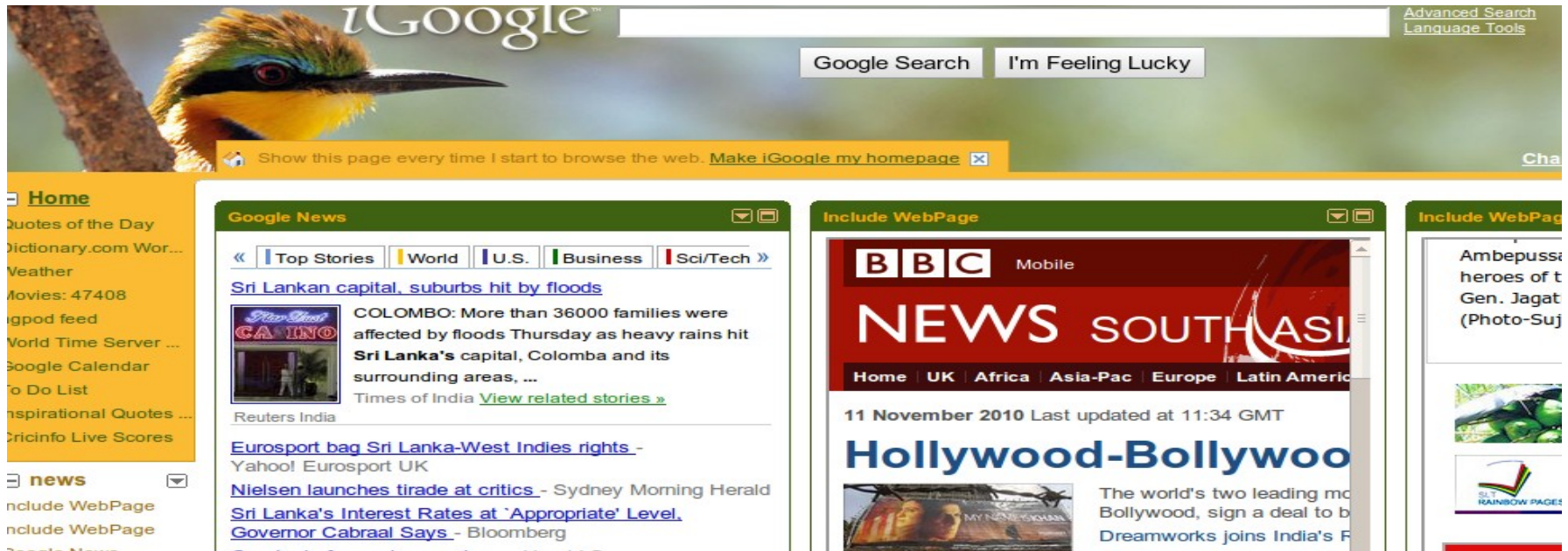
- Web 2.0, Gadgets, and Gadgets Communication
- Tuples and Gadgets
- Proposed Solution: In browser vs. Global tuple space
- Architecture
- Evaluation
- Usecases



- User generated or user extended content and collective intelligence are the key
- Need to be simple, sensible, and scalable
- Users add value, often as a side effect
- Data is your competitive advantage
- Mahups/ Remix



Web Gadgets



- Common way to generate Web 2.0 Content
- Almost its own page, which is rendered in its own iframe.
- It is described through gadget.xml
- Rendered by a gadget container (e.g. iGoogle and Shindig)
- Let users customize their view (profile) by adding removing features they like to see.

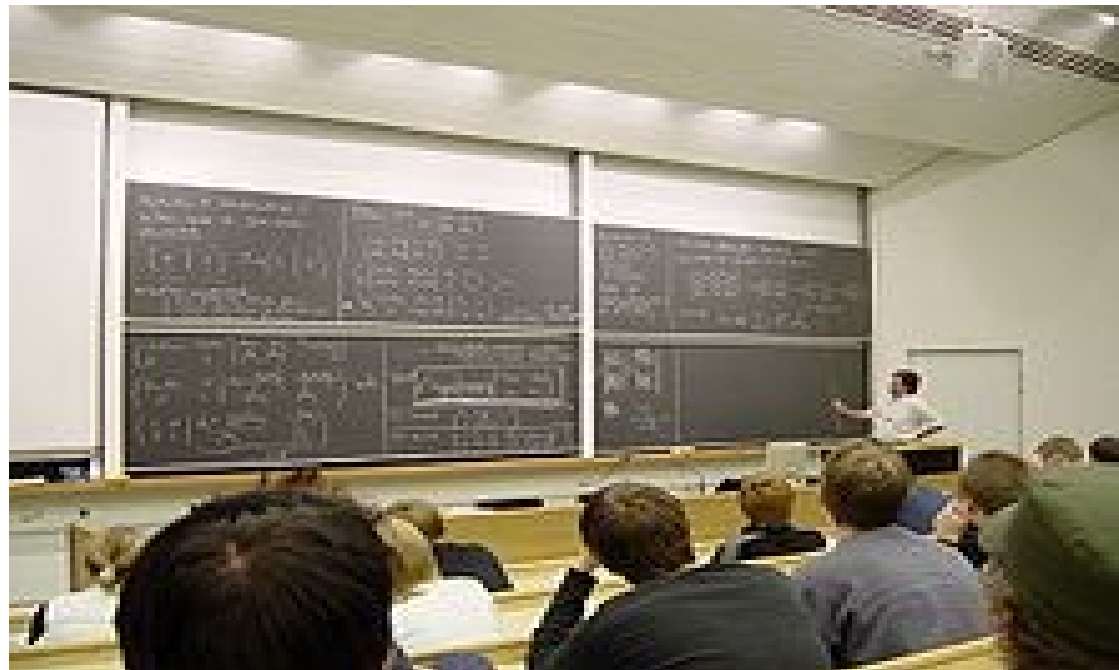
Next generation Gadgets/ Apps?

- Enable truly client side applications (break the need to put code to backend) – e.g. Online games, social network apps
 - Inter Gadget communications?
 - To store settings, client side communication, communication between many instances of the same gadget or different gadgets.
- How to do that?
 - Pub/Sub – pubsubhubbub
 - RPC – call services in the backend
 - By navigating documents - if you know about the other Gadget, you can access it though the DOM
 - Google Gears (Only local)



Tuple Spaces

- Back to school: all of us know what a Tuple is; $\langle \text{Nimal}, 23, \text{central college}, \rangle$
- Tuple space is a backboard, shared place to
 - Put tuples
 - Search and read tuples
 - Search and retrieve (and remove tuples)
- Loosely coupled
- An implementation of the associative memory paradigm for parallel/distributed computing.
- A repository of Tuples that can be accessed concurrently by one or more processors.





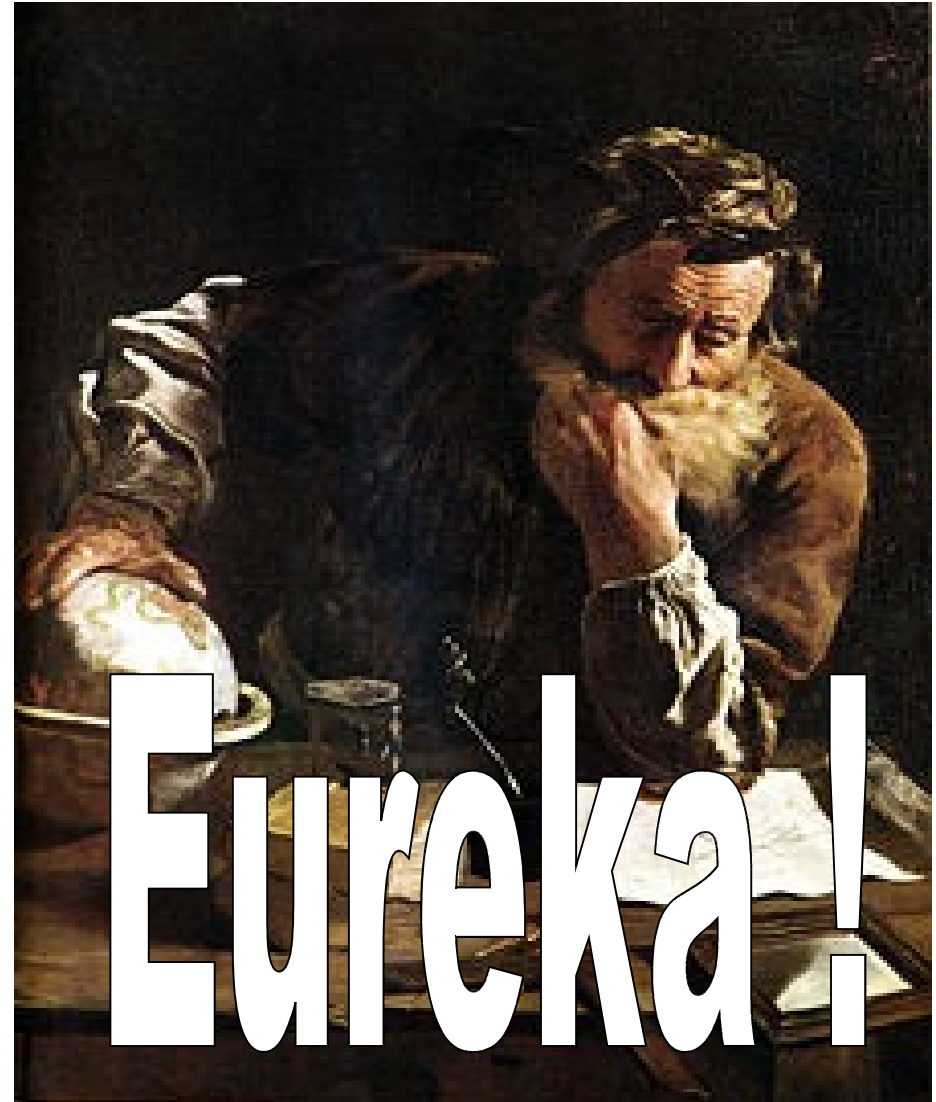
Tuple Spaces for Gadget Communication and Data Storage

- Loose Coupling between entities
 - Decoupling in time – do not need both side to be online
 - Decouple in space – do not need direct addressing
 - Decouple in synchronization - Async matches java script model
- Stable storage (replicated, scalable, and reliable)
- Support for search through wild cards
- Both Push and Pull models to get data.

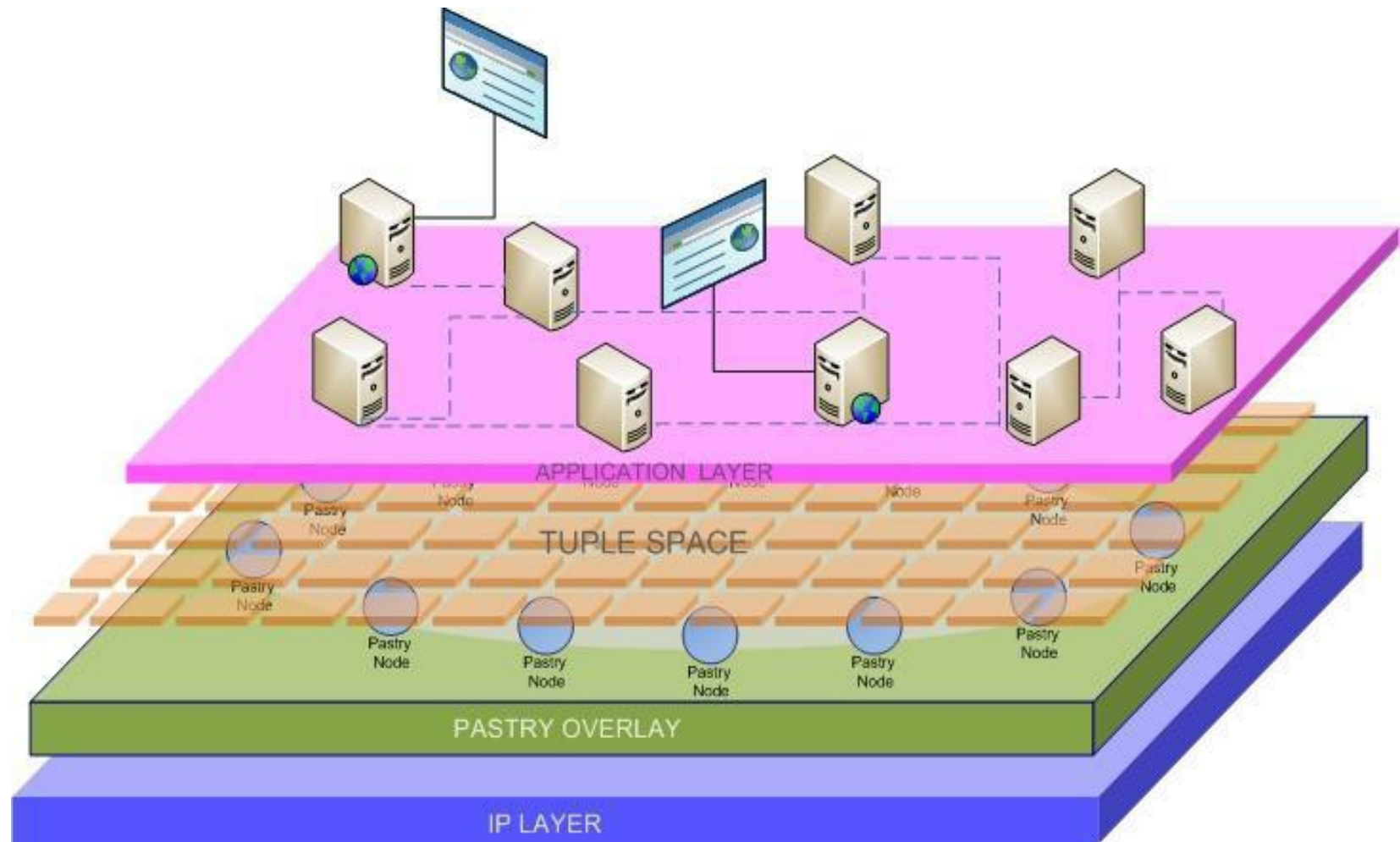
Proposed Solution

Two parts

- We proposed to use a Scalable Tuple space to share state and to communicate across Web applications
- We have built a in browser tuple space and an external Global tuple space, and link them together



Part1: Global Tuple Space



- BISSA implements a distributed tuple space based on peer to peer paradigm (On top of Pastry)
- Highly scalable and reliable

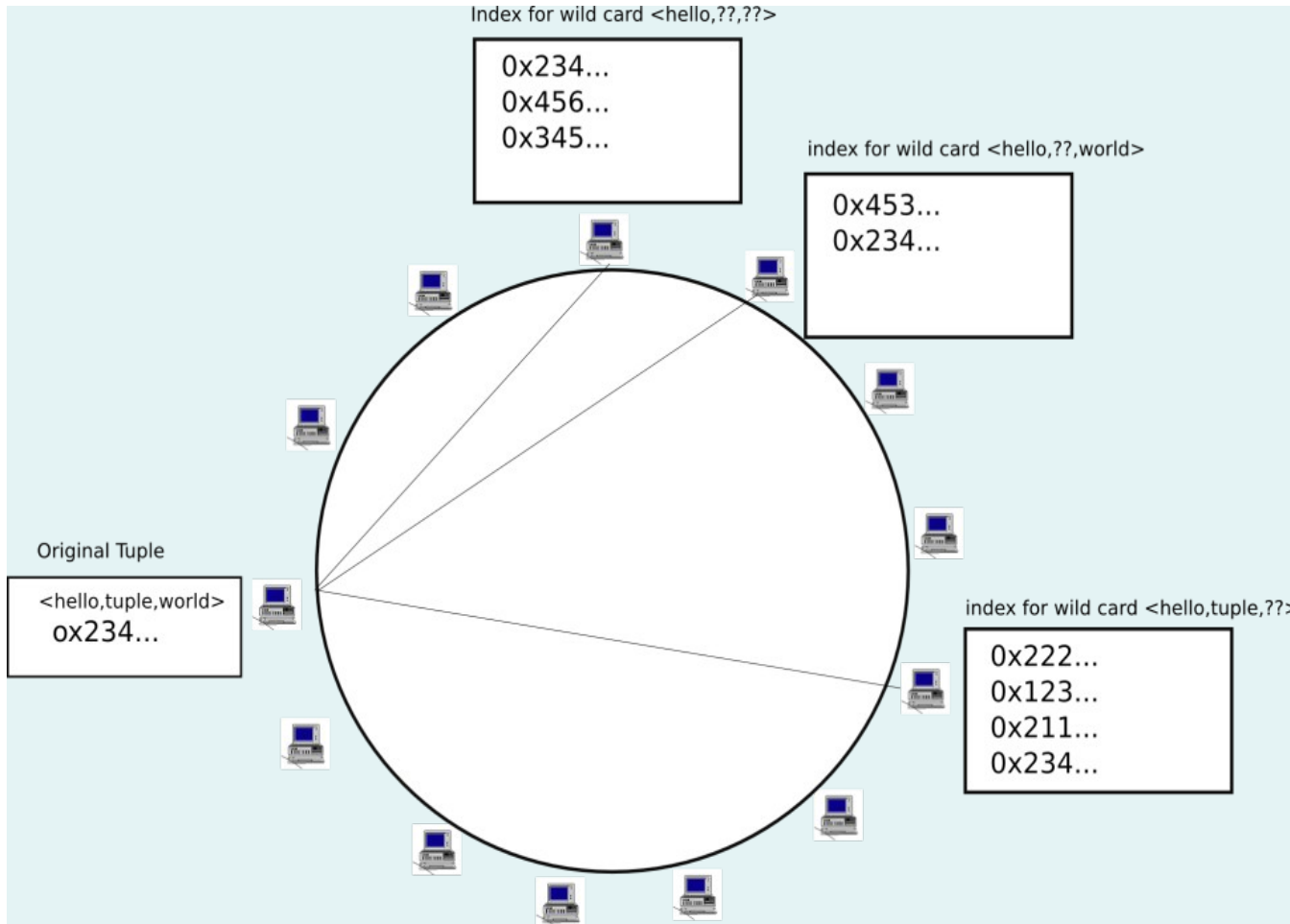


Global Space: Challenge

- Idea is to store tuples in DHT using DHT replication mechanisms
- DHT let us distribute the data across nodes, thus scales very well.
- Challenge is to support search based on templates (e.g. $\langle \text{hello}, \text{????} \rangle$) for “read”, “take”, and “subscribe” operations.

Big Idea: Generate templates for each value and build an index. (e.g. for each $\langle a, b \rangle$ create $\langle a, * \rangle$ and $\langle *, b \rangle$ templates and build a index for each).

Architecture: Supporting Write

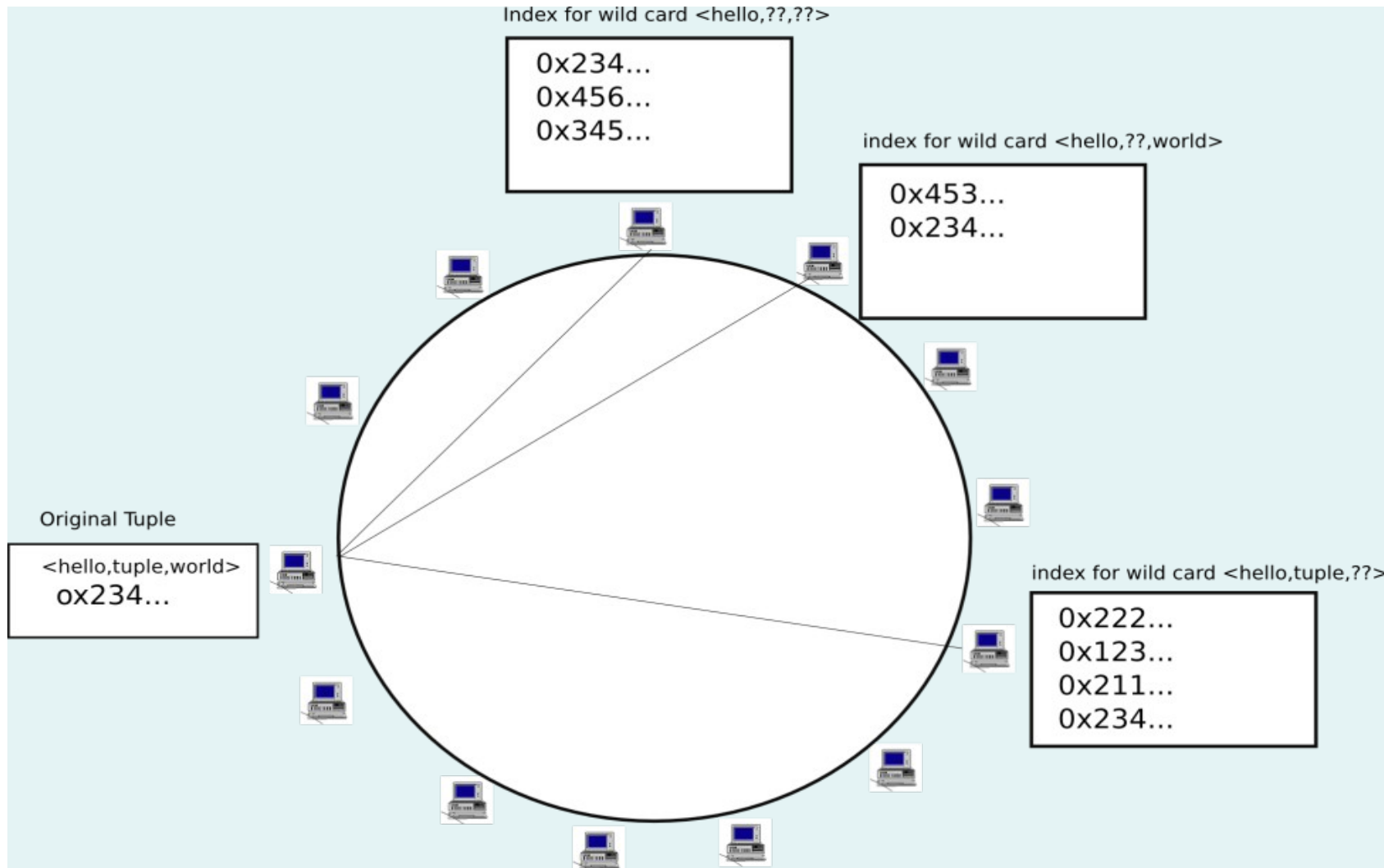


Two main phases

- 1) Write tuple as a DHT element with an associated hash calculated for it.
- 2) Update the Indexes that are distributed in the system.

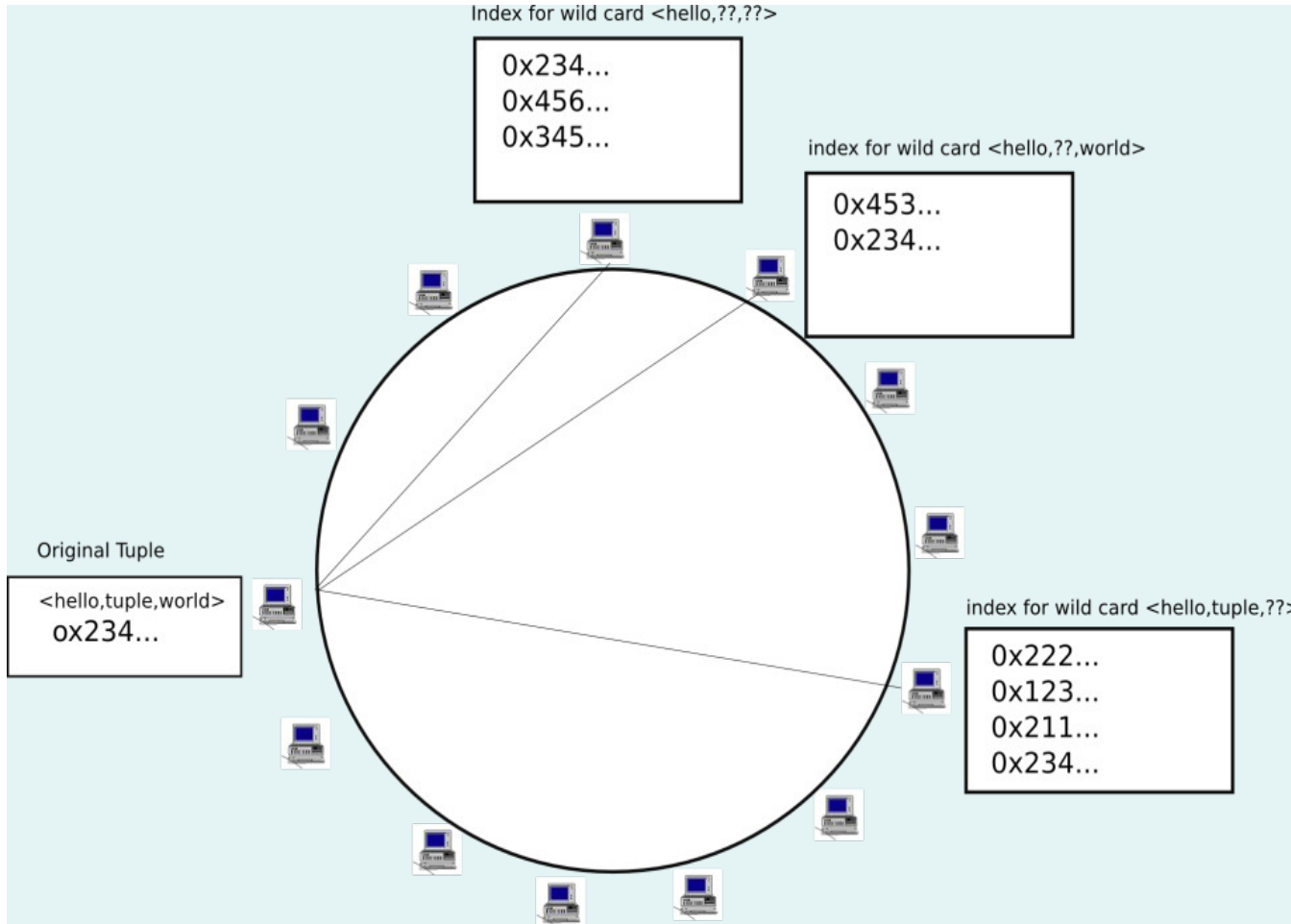
Search is done through Indexes associated with tuple templates. Ex : tuple <hello,world> can have <???,world> or <hello,???,>, and there is a index for each template.

Architecture: Supporting Read



- Hash the search pattern and lookup the index for that pattern using DHT. Then lookup entries for that pattern using index.

Architecture: Supporting Take

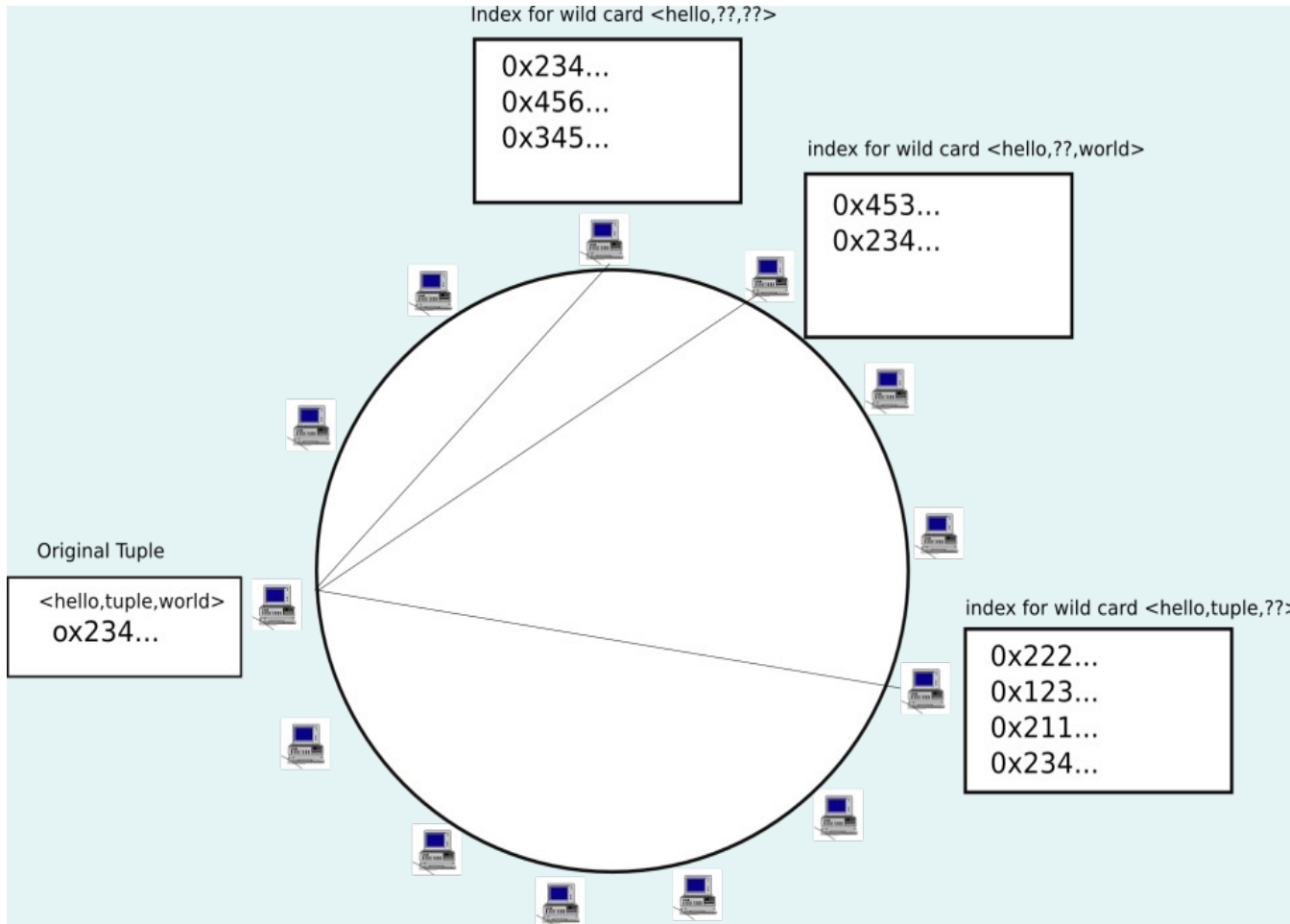


Three main phases

- 1) Get the tuple locations that matches the pattern using index
- 2) Delete the actual tuple(s)
- 3) Update the indexes to not have the tuples

Only support sync manner

Architecture: Supporting Subscriptions



How?

- 1) Store the subscriptions with indexes.
- 2) When index is being updated, notify the subscribers.

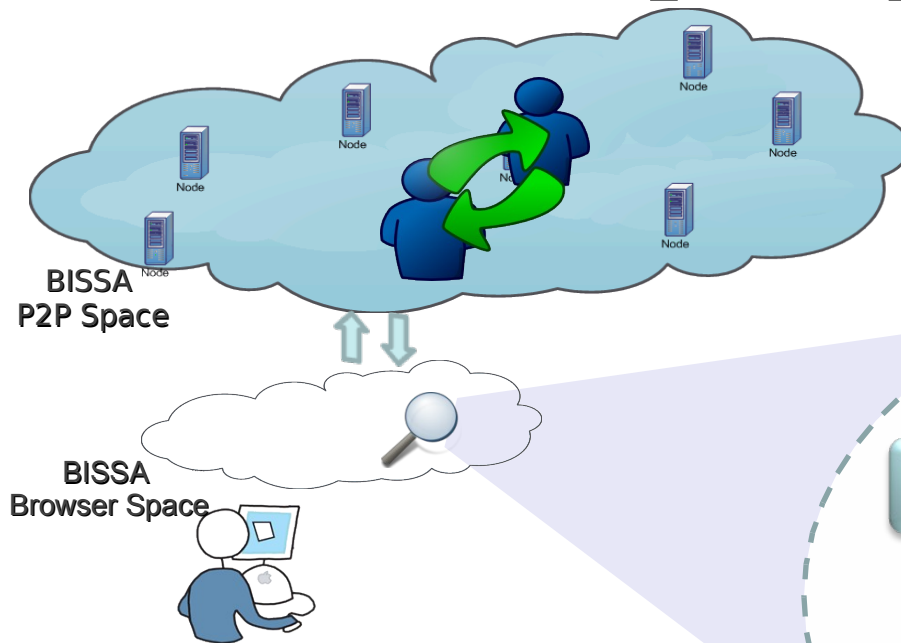
Subscriptions are also done using patterns



Consistency Model

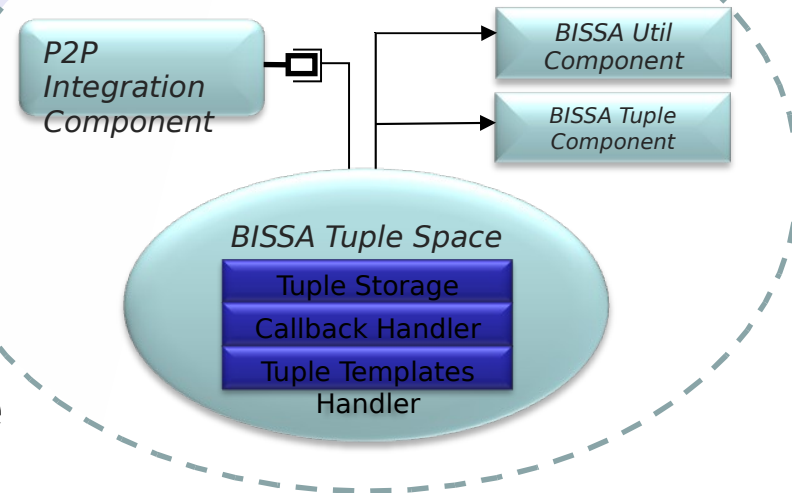
- In the take operation, updating indexes and updating entries are not atomic. So values are inconsistent while take is happening.
- We first delete the entries then delete the index
- What if two takes or take and read happened at the same time?
 - If replicas in active passive mode, we can try to detect and handle this at the API level. (e.g. If tuples are removed while retrieving them after a search, that suggests take is going on.
 - Otherwise we have eventual consistency - OK for some Apps like distributed Computation, Gaming Apps

In-Browser Tuple Space



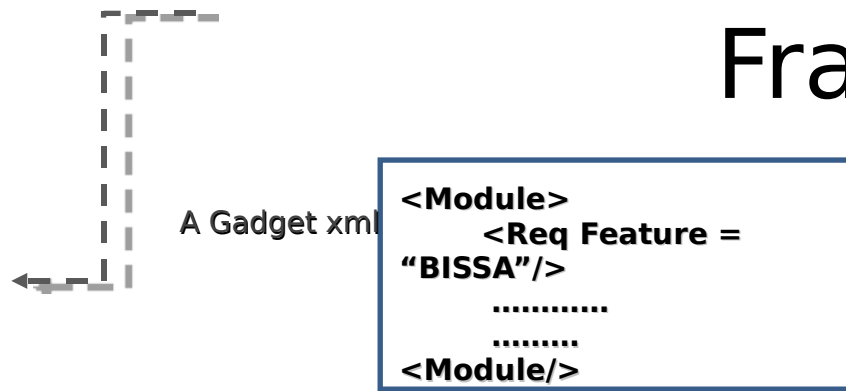
Operation

- insert Tuples → put()
- query Tuple → query()
- remove Tuples → take()

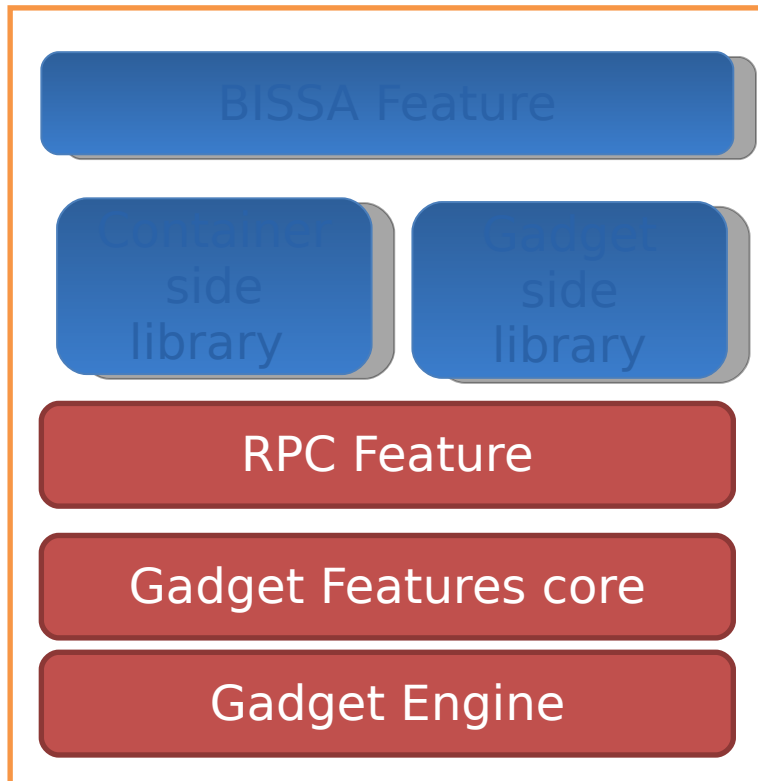


- In memory browser tuple space
- Local APIs and Global APIs

BISSA Inter Gadget Communication Framework



- Develop a Tuple space in java scripts
- JavaScript Library for gadget level Access for BISSA
- Included this as a Gadget Feature through Shinding: Can get access to Bissa libraries by adding `<Req feature="Bissa" />`
- Can talk to local API or global API
- Access to global API happens through RPC bridge.
- We sync the local and global tuple spaces periodically. WS for communication



How does it make a difference?

- Enable truly client side applications: break the need to put code to backend, and now it is possible to develop Apps that communicates with each other and have storage support without any backend code.
 - Key to make authoring easier
- Provide coordination between application in a loosely coupled unlike other server based solutions
- Highly scalable due to underline DHT
- Reliability through replication
- Storage - Remove the need to have and maintain a database
- Provides Pub/sub with support for data persistence as well.

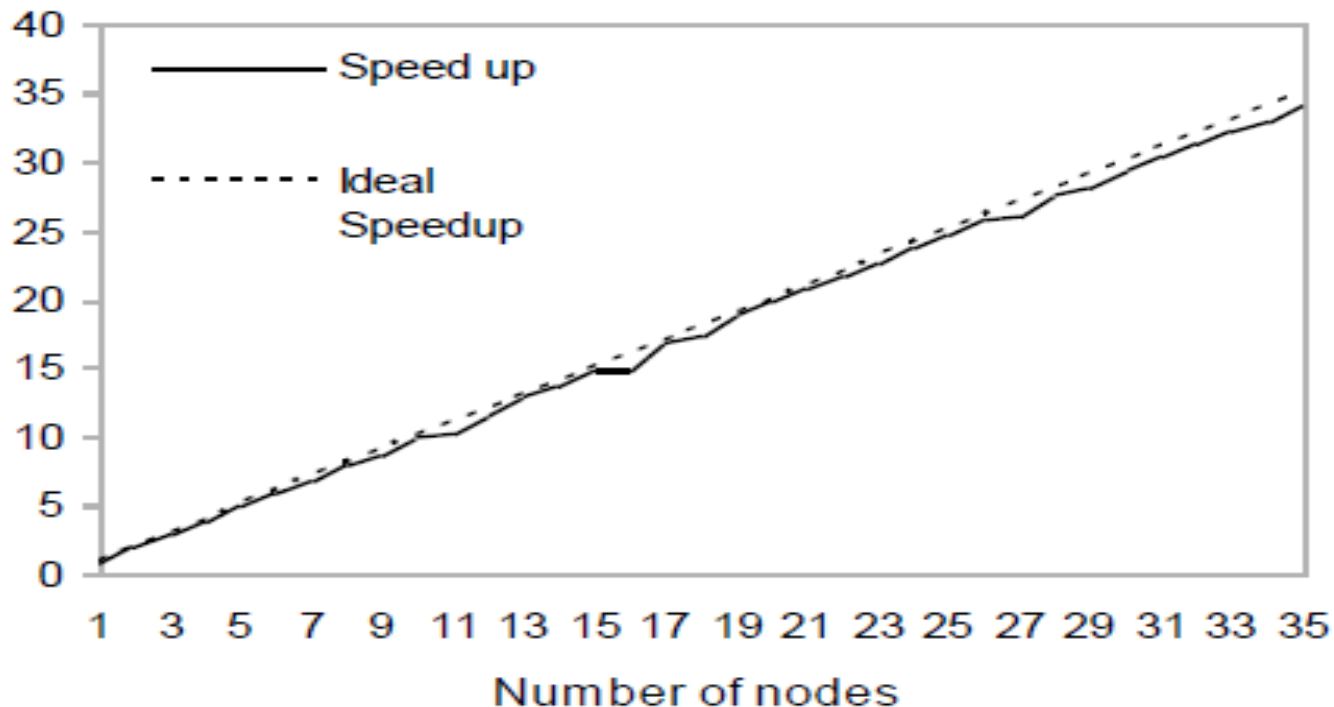




Few Use Cases

- Co-ordination among gadget based applications
 - e.g. Shared data dashboard: One App can publish current weather forecasts which is used by other Apps.
 - Games: Two players play chess by coordinating through Bissa
 - Social Apps : find out what happen with friends through Bissa (e.g. Implement open social with Bissa)
- Highly scalable & available data storage for web applications
 - Remember user preferences
 - Distributed Address book
- As a Pub Sub framework for web Application
- CPU scavenging using web browsers
 - bringing processing power of browsers in to the community grid
- Distributed Cache for Web Applications

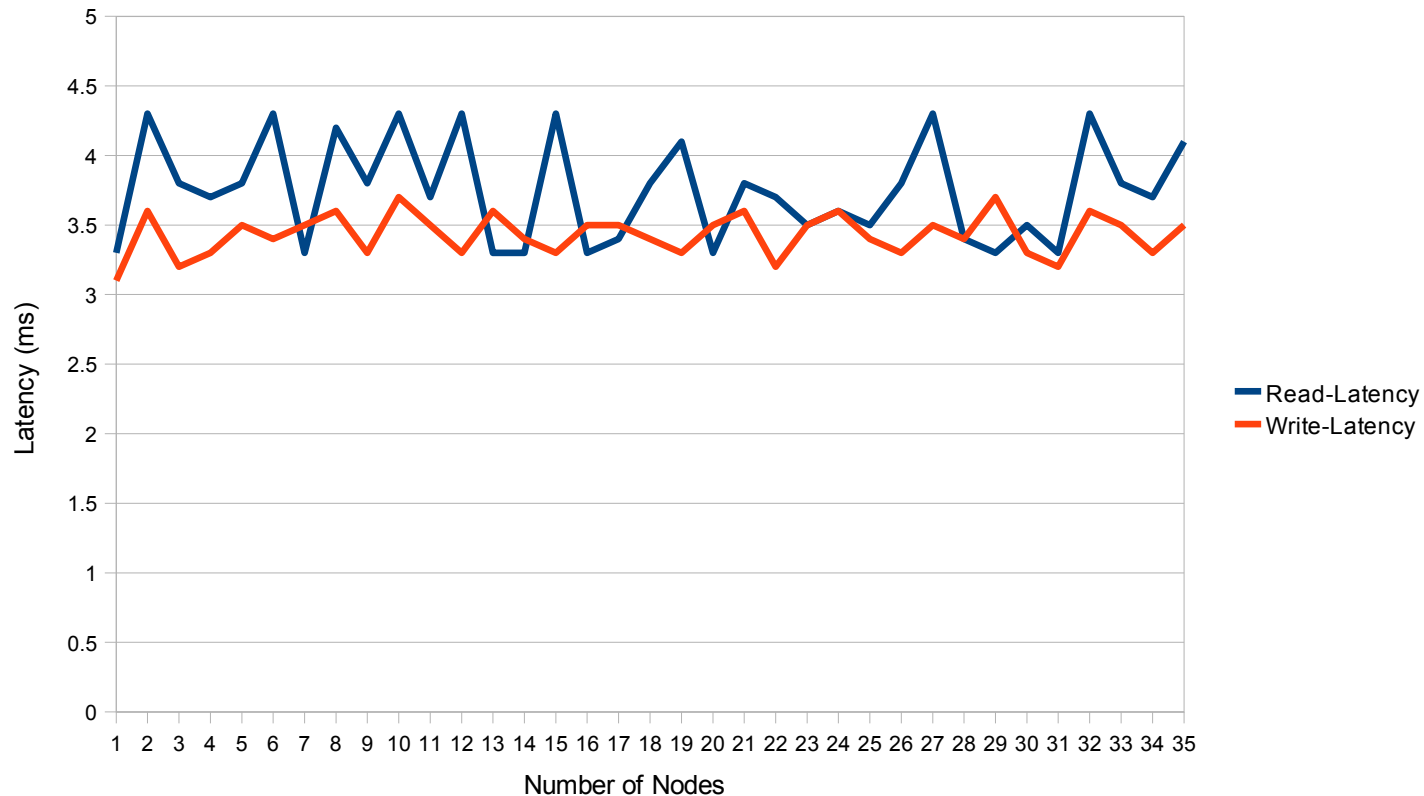
Testing Global Space: Scalability Test



A scale test was conducted using 35 computers in a controlled lab environment

Used “embarrassingly parallel” Monte-Carlo simulation as the testing mechanism

Testing Global Space: Latency Test



Measured the operation latency for constant number of data in space while system is scaling up
Kato and Kamia [14] shows that the messaging latency of Pastry is quite good up to 800 nodes



Testing Browser Space

Operation	Latency
read --> 30ms	30ms
write	35ms
Take	210ms

- Value is much higher than accessing from P2P ring
- We believe it is caused by async nature of http request from browser and cost for WS call
- Plan to explore this more.



Future Work

- Testing Scalability
- Further understanding in to consistency model
- Security model for Bissa
- Further refine it for distributed cache
- Avoid polling from local to global state through Web Sockets



Summary

- Web Gadgets And Apps
- Next Generation Gadgets/ Apps with storage and communications
- Global space, how it work?
- Local space How it works?
- Usecases and Impact
- Performance measurements



Q&A

Thank You!!!



Project web site (<http://bissa.sourceforge.net/>) is hosted at sourceforge.net, & has every resources for developers including blogs/samples/manuals & screen-casts.

The BISSA code is licensed under Apache 2.0 license apart from Freepastry license : BSD license making it a 100% open source project.