Imperial College London

LSDS Large-Scale Distributed Systems Group

Bridging the Gap between Serving and Analytics in Scalable Web Applications

Panagiotis Garefalakis

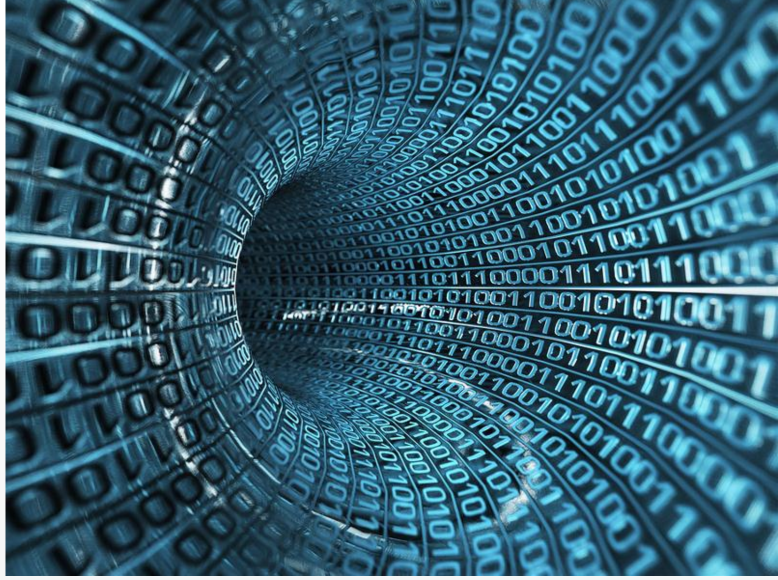
M.Res Thesis Presentation, 7 September 2015

play2sdg

Outline

- Motivation
- Challenges
    - ➡Scalable web app design
    - ➡Resource efficiency
    - ➡Resource Isolation
- In-memory Web Objects model
    - ➡Play2SDG case study
    - ➡Experimental Results
- Conclusions
- Future work

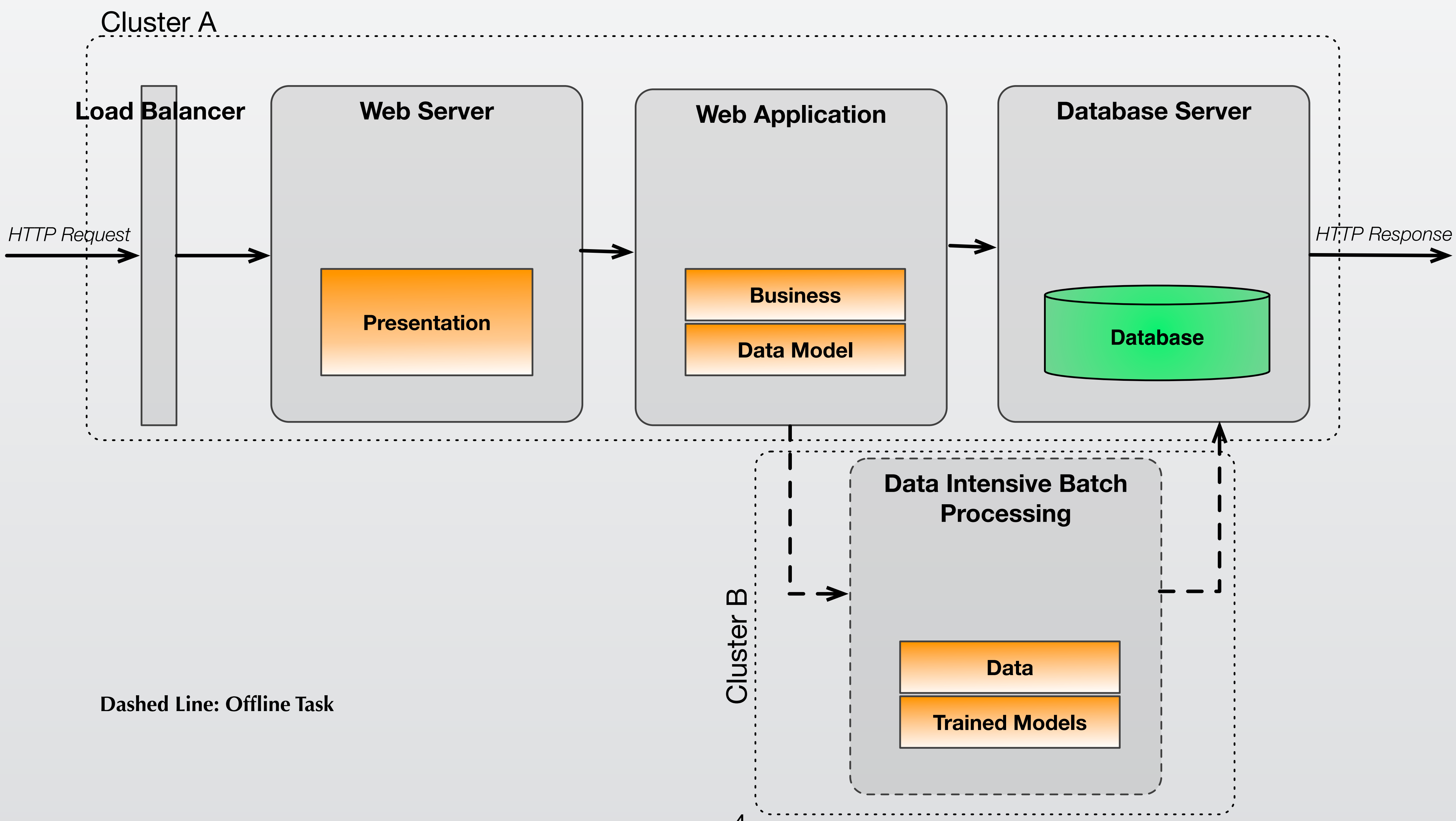
Motivation



- Most modern web and mobile applications today offer highly **personalised** services generating **large** amounts of data
- Tasks separated into **offline** (BE) and **online** (LC) based on the latency, computation and data freshness requirements
- To train models and offer **analytics**, they use asynchronous offline computation, which leads to stale data being served to clients
- To **serve** requests robustly and with low latency, applications cache data from the analytics layer
- Applications deployed in **large clusters**, but with no collocation of tasks to avoid SLO violations
- No data **freshness** guarantees and poor resource **efficiency**

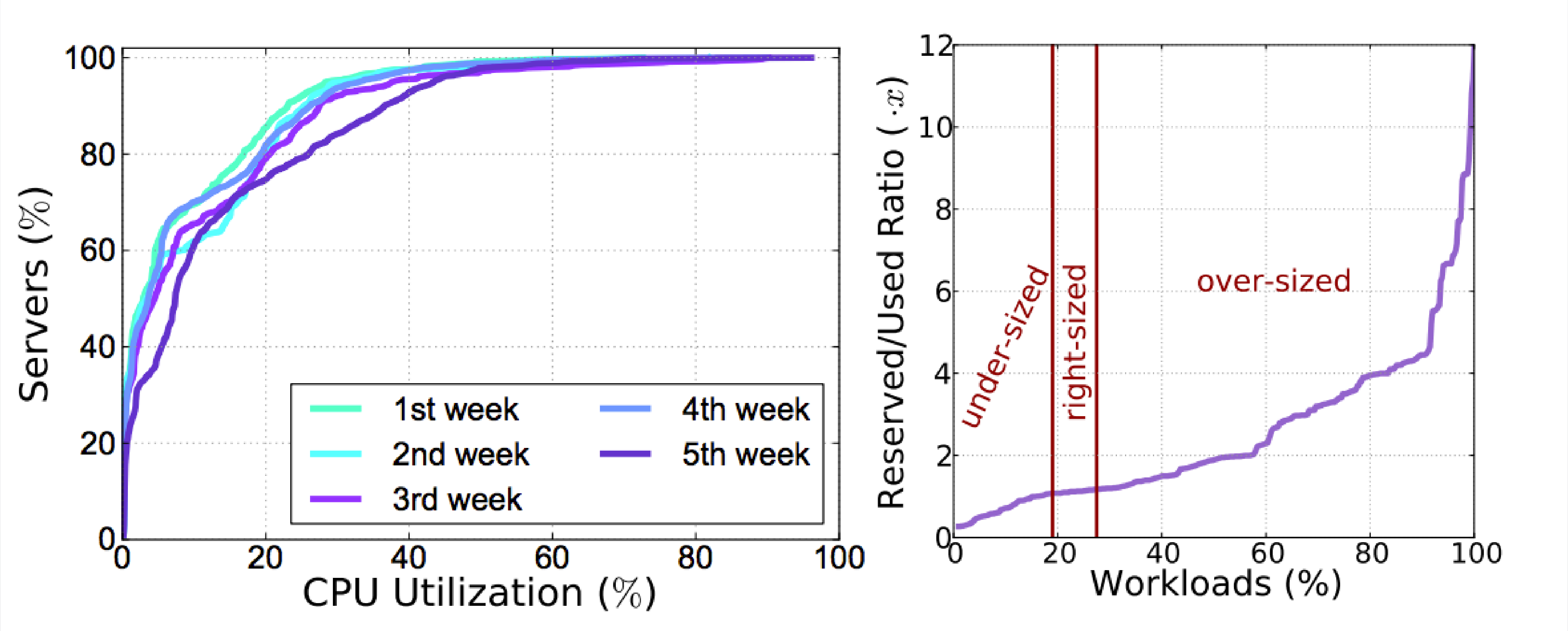
Typical Web App

- How does a typical scalable web application look like?
- There is a strict decoupling of online and offline tasks
- With the emerge of cloud computing, these applications are deployed on clusters with thousands of machines



Challenges: Resource Efficiency

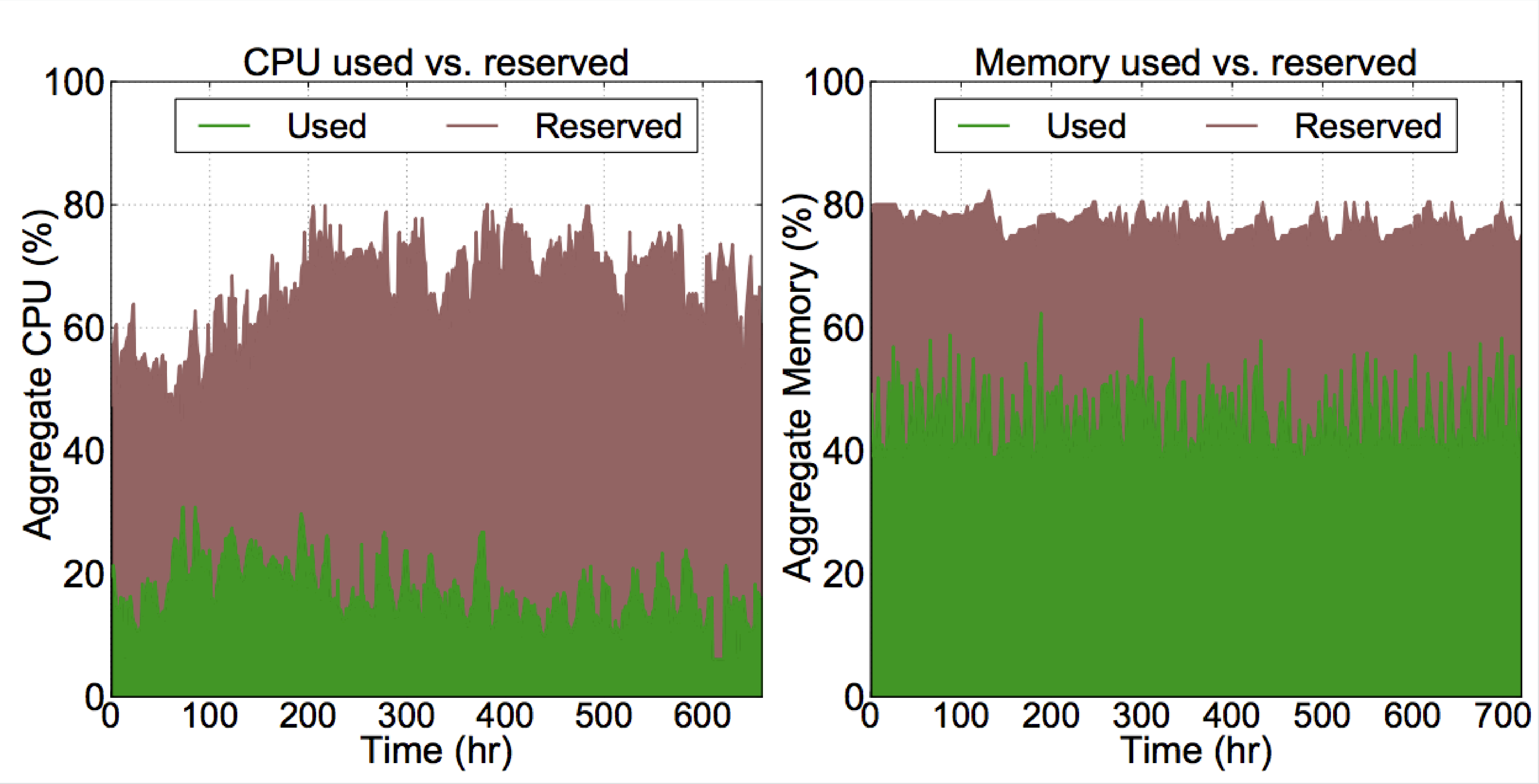
- Most cloud facilities operate at very low utilisation, hurting both cost effectiveness and future scalability
- Figure depicts a utilisation analysis for a production cluster at Twitter with thousands of servers, managed by Mesos over one month. The cluster mostly hosts user-centric services
- The aggregate CPU utilisation is consistently below 20%, even though reservations reach up to 80% of total capacity



Delimitrou, Christina, and Christos Kozyrakis. "Quasar: Resource-efficient and qos-aware cluster management. ASPLOS 2014

Challenges: Resource Efficiency

- Even when looking at individual servers, their majority does not exceed 50% utilisation on any week
- Typical memory use is higher (40-50%) but still differs from the reserved capacity



Delimitrou, Christina, and Christos Kozyrakis. "Quasar: Resource-efficient and qos-aware cluster management. ASPLOS 2014

Imperial College London

Challenges: Resource Isolation

- Shared cluster environments suffer from resource **interference**. The main resources that are affected are CPU, caches (LLC), memory (DRAM), and network. There are also non-obvious interactions between resources, known as cross-resource interactions
- What about resource isolation mechanisms provided by the Operating System - through scheduling?
- Even at low load, colocating LC with BE tasks creates sufficient pressure on the shared resources to lead to SLO violations. There are differences depending on the LC sensitivity on shared resources
- The values are latencies, normalised to the SLO latency

websearch

	5%	10%	15%	20%	25%	30%	35%	40%	45%	50%	55%	60%	65%	70%	75%	80%	85%	90%	95%
LLC (small)	134%	103%	96%	96%	109%	102%	100%	96%	96%	104%	99%	100%	101%	100%	104%	103%	104%	103%	99%
LLC (med)	152%	106%	99%	99%	116%	111%	109%	103%	105%	116%	109%	108%	107%	110%	123%	125%	114%	111%	101%
LLC (big)	>300%	>300%	>300%	>300%	>300%	>300%	>300%	>300%	>300%	>300%	>300%	>300%	>300%	>300%	>300%	264%	222%	123%	102%
DRAM	>300%	>300%	>300%	>300%	>300%	>300%	>300%	>300%	>300%	>300%	>300%	>300%	>300%	>300%	>300%	270%	228%	122%	103%
HyperThread	81%	109%	106%	106%	104%	113%	106%	114%	113%	105%	114%	117%	118%	119%	122%	136%	>300%	>300%	>300%
CPU power	190%	124%	110%	107%	134%	115%	106%	108%	102%	114%	107%	105%	104%	101%	105%	100%	98%	99%	97%
Network	35%	35%	36%	36%	36%	36%	36%	37%	37%	38%	39%	41%	44%	48%	51%	55%	58%	64%	95%
brain	158%	165%	157%	173%	160%	168%	180%	230%	>300%	>300%	>300%	>300%	>300%	>300%	>300%	>300%	>300%	>300%	>300%

ml_cluster

	5%	10%	15%	20%	25%	30%	35%	40%	45%	50%	55%	60%	65%	70%	75%	80%	85%	90%	95%
LLC (small)	101%	88%	99%	84%	91%	110%	96%	93%	100%	216%	117%	106%	119%	105%	182%	206%	109%	202%	203%
LLC (med)	98%	88%	102%	91%	112%	115%	105%	104%	111%	>300%	282%	212%	237%	220%	220%	212%	215%	205%	201%
LLC (big)	>300%	>300%	>300%	>300%	>300%	>300%	>300%	>300%	>300%	>300%	>300%	>300%	>300%	>300%	276%	250%	223%	214%	206%
DRAM	>300%	>300%	>300%	>300%	>300%	>300%	>300%	>300%	>300%	>300%	>300%	>300%	>300%	>300%	>300%	287%	[illegible]	[illegible]	[illegible]
HyperThread	113%	109%	110%	111%	104%	100%	97%	107%	111%	112%	114%	114%	114%	119%	121%	130%	[illegible]	[illegible]	[illegible]
CPU power	112%	101%	97%	89%	91%	86%	89%	90%	89%	92%	91%	90%	89%	89%	90%	92%	[illegible]	[illegible]	[illegible]
Network	57%	56%	58%	60%	58%	58%	58%	58%	59%	59%	59%	59%	59%	63%	63%	67%	[illegible]	[illegible]	[illegible]
brain	151%	149%	174%	189%	193%	202%	209%	217%	225%	239%	>300%	>300%	279%	>300%	>300%	>300%	[illegible]	[illegible]	[illegible]

memkeyval

	5%	10%	15%	20%	25%	30%	35%	40%	45%	50%	55%	60%	65%	70%	75%	80%	[illegible]	[illegible]	[illegible]
LLC (small)	115%	88%	88%	91%	99%	101%	79%	91%	97%	101%	135%	138%	148%	140%	134%	150%	[illegible]	[illegible]	[illegible]
LLC (med)	209%	148%	159%	107%	207%	119%	96%	108%	117%	138%	170%	230%	182%	181%	167%	162%	[illegible]	[illegible]	[illegible]
LLC (big)	>300%	>300%	>300%	>300%	>300%	>300%	>300%	>300%	>300%	>300%	>300%	>300%	>300%	280%	225%	222%	[illegible]	[illegible]	[illegible]
DRAM	>300%	>300%	>300%	>300%	>300%	>300%	>300%	>300%	>300%	>300%	>300%	>300%	>300%	>300%	252%	234%	[illegible]	[illegible]	[illegible]
HyperThread	26%	31%	32%	32%	32%	32%	33%	35%	39%	43%	48%	51%	56%	62%	81%	119%	[illegible]	[illegible]	[illegible]
CPU power	192%	277%	237%	294%	>300%	>300%	219%	>300%	292%	224%	>300%	252%	227%	193%	163%	167%	[illegible]	[illegible]	[illegible]
Network	27%	28%	28%	29%	29%	27%	>300%	>300%	>300%	>300%	>300%	>300%	>300%	>300%	>300%	>300%	[illegible]	[illegible]	[illegible]
brain	197%	232%	>300%	>300%	>300%	>300%	>300%	>300%	>300%	>300%	>300%	>300%	>300%	>300%	>300%	>300%	[illegible]	[illegible]	[illegible]

Each entry is color-coded as follows: 140% is ≥120%, 110% is between 100% and 120%, and 65% is ≤100%.

When a number of workloads execute concurrently on a server, they compete for shared resources.

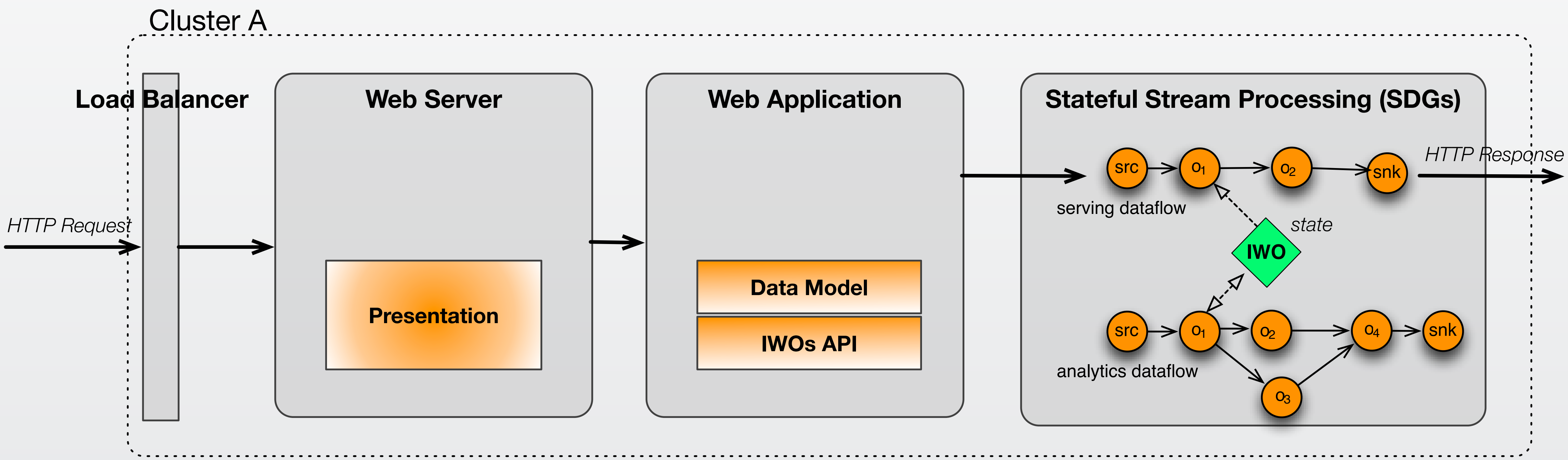
Heracles: Improving Resource Efficiency at Scale," David Lo, Liqun Cheng, Rama Govindaraju, Parthasarathy Ranganathan, and Christos Kozyrakis. ISCA 2015.

play2sdg

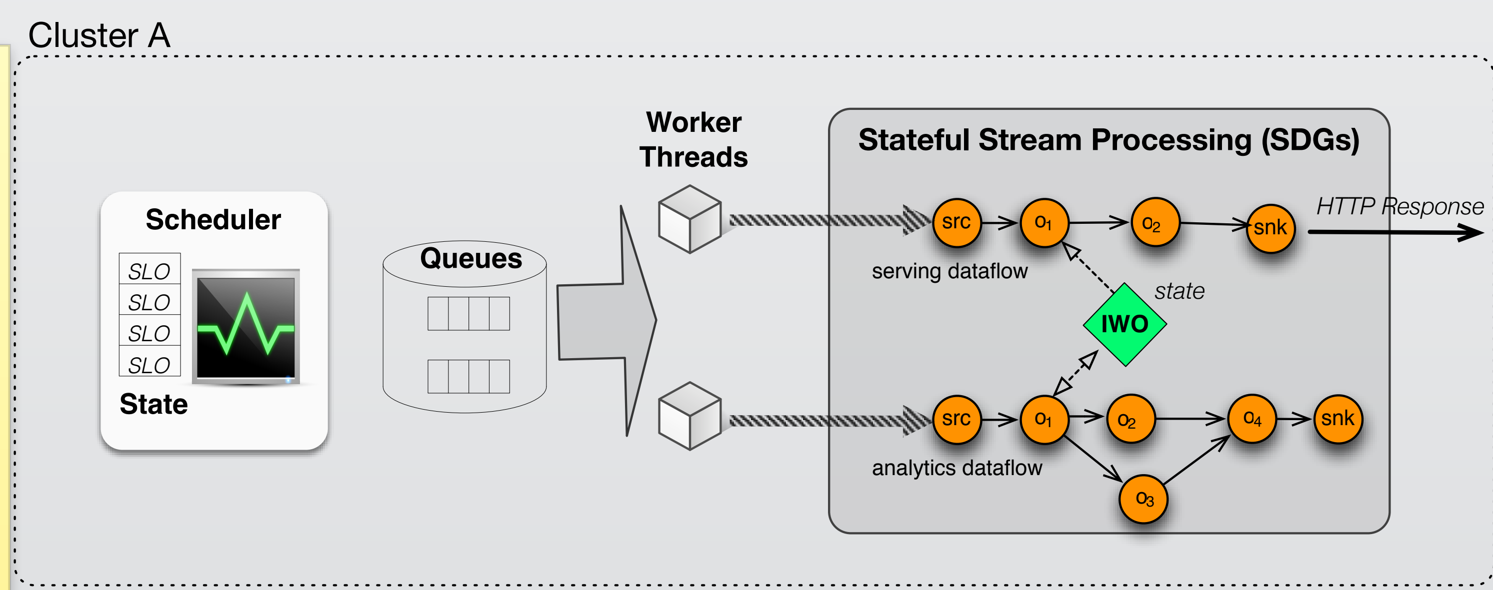
Imperial College London

In-memory Web Objects Model

- **IWOs,** express both online and offline logic of a web application as a single stateful distributed dataflow graph (SDG)
- **State** of the dataflow computation is expressed as IWOs, which are accessible as persistent objects by the application
- What about application strict **SLOs** - resource isolation and efficiency?



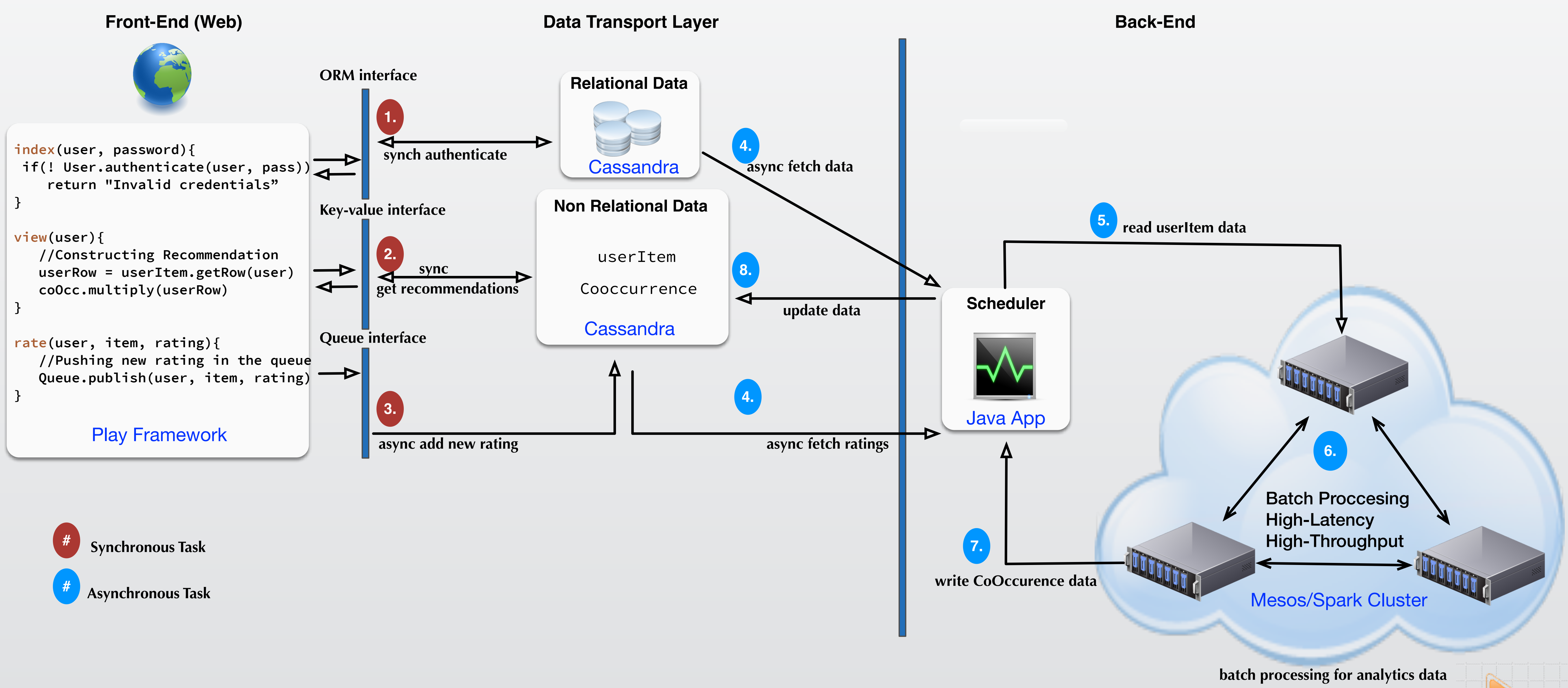
Tasks can be cooperatively scheduled, allowing to move resources between tasks of the dataflow efficiently according to the web application needs. As a result, the application can exploit data-parallel processing for compute-intensive requests and also maintain high resource utilisation, e.g. when training complex models, leading to fresher data while serving results with low latency from IWOs.



play2sdg

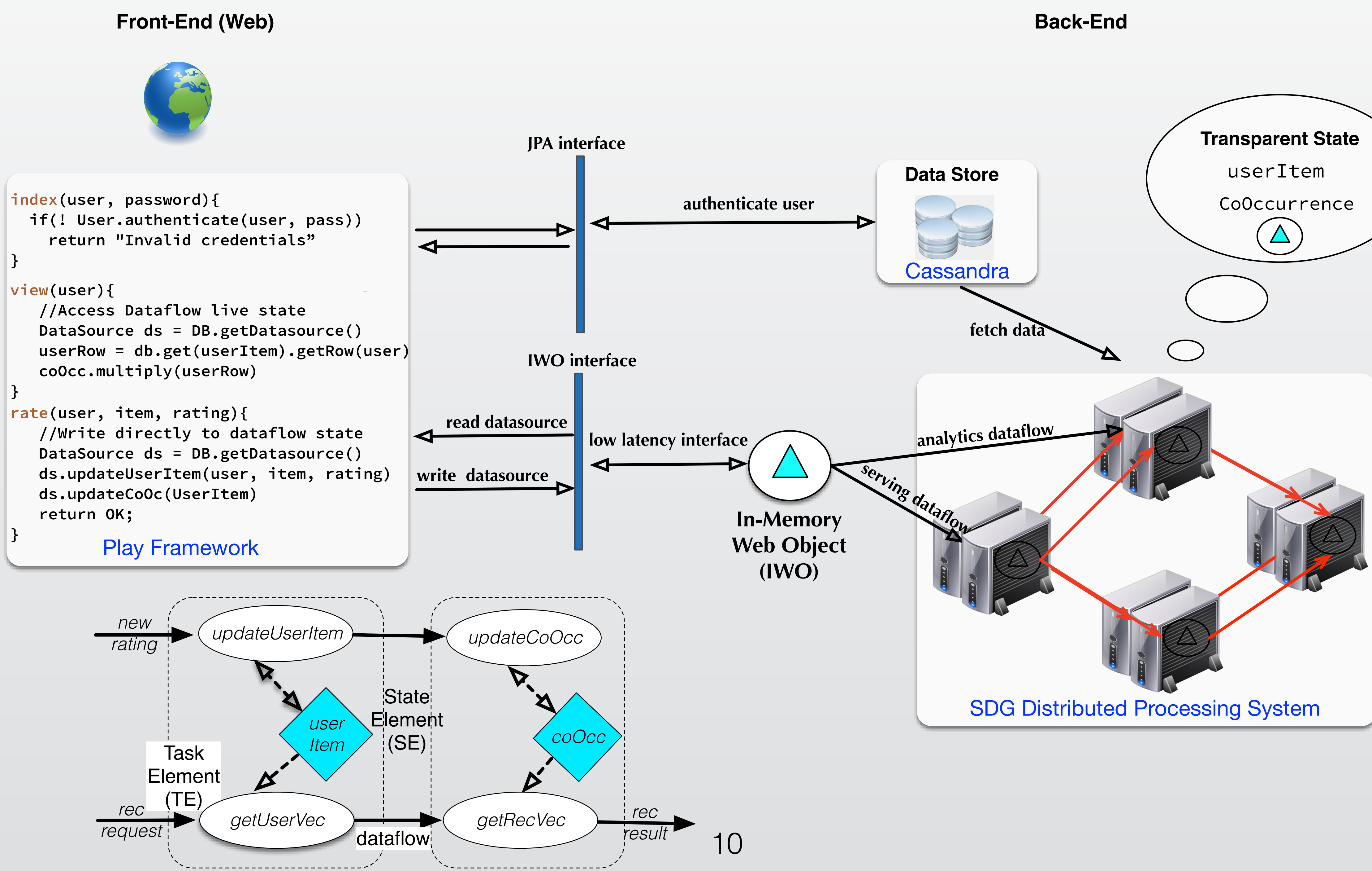
Play2SDG: Typical Web Music App

- Implemented a typical scalable web music service using Play Framework for Java
- **Decoupled** online and offline tasks to lower response latency
- **Asynchronous** collaborative filtering (CF) task using Apache Spark and Mesos for deployment



Play2SDG: IWOs Web Music App

- Implemented a scalable web music service using **IWOs API** and making minor changes in the application code
- Express both online and offline logic of a web application as a stateful distributed dataflow graph
- Online collaborative filtering implementation using SDGs. addRating must achieve high throughput; getRec must serve requests with low latency, when recommendations are included in dynamically generated web pages



play2sdg

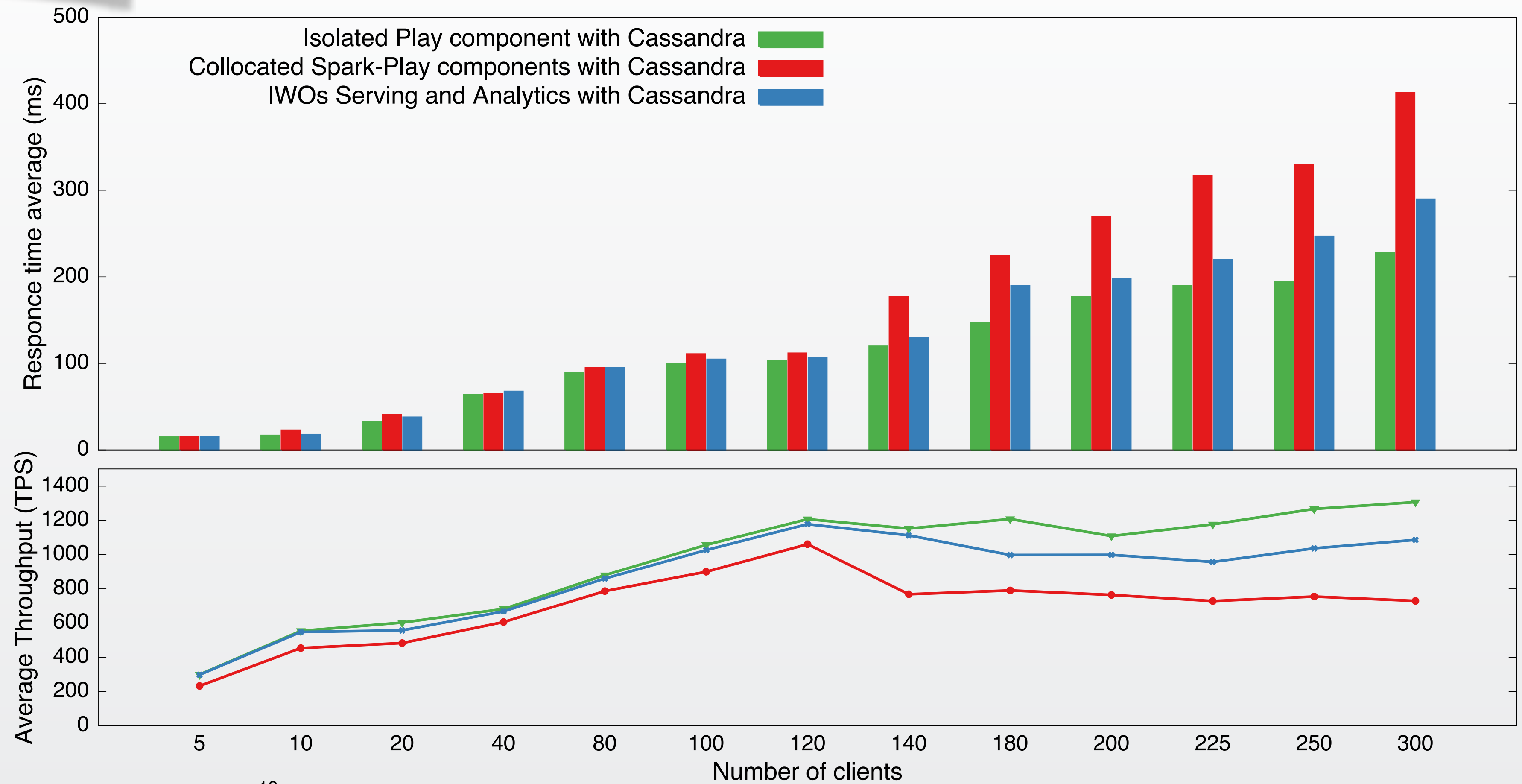
Evaluation Platform

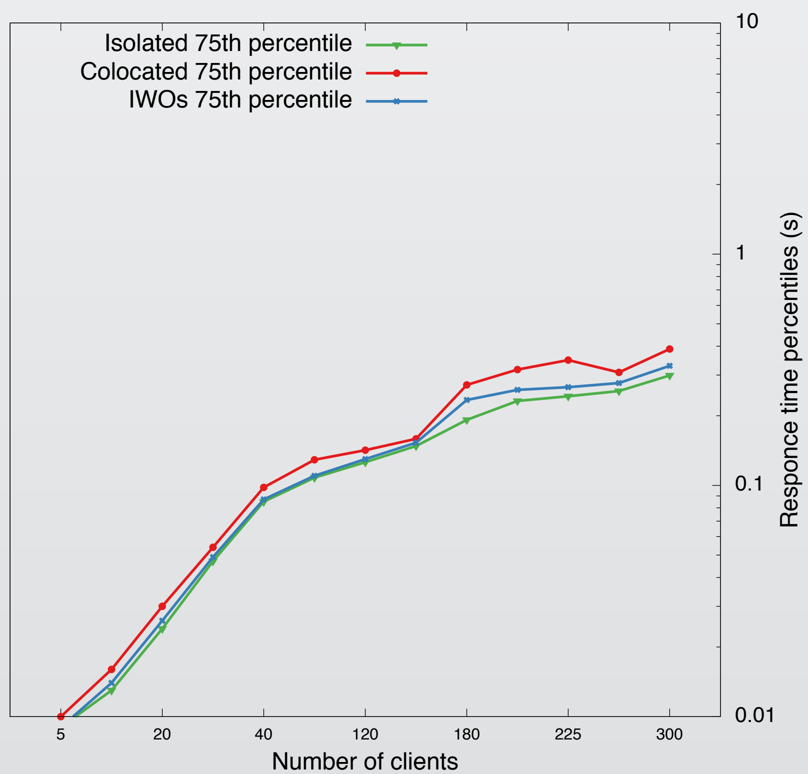
- Wombat's private cluster with 5 machines
- Machines with 8 CPUs, 8 GB RAM and 1TB locally mounted disk, 1Gbps network
- Data: Million song Dataset, 943,347 unique tracks with 8,598,630 tag pairs
- **Software**:
  - Apache Spark 1.1.0
  - Apache Mesos is 0.22.1 (1 master node 3 slaves)
  - Nginx is 1.1.19
  - Cassandra database is 2.0.1
- **Load generator** is Apache JMeter 2.13 producing a specific functional behaviour pattern:
  1. user login
  2. navigate through the home page displaying the top 100 tracks
  3. visit the page with the latest recommendations
  4. user logout

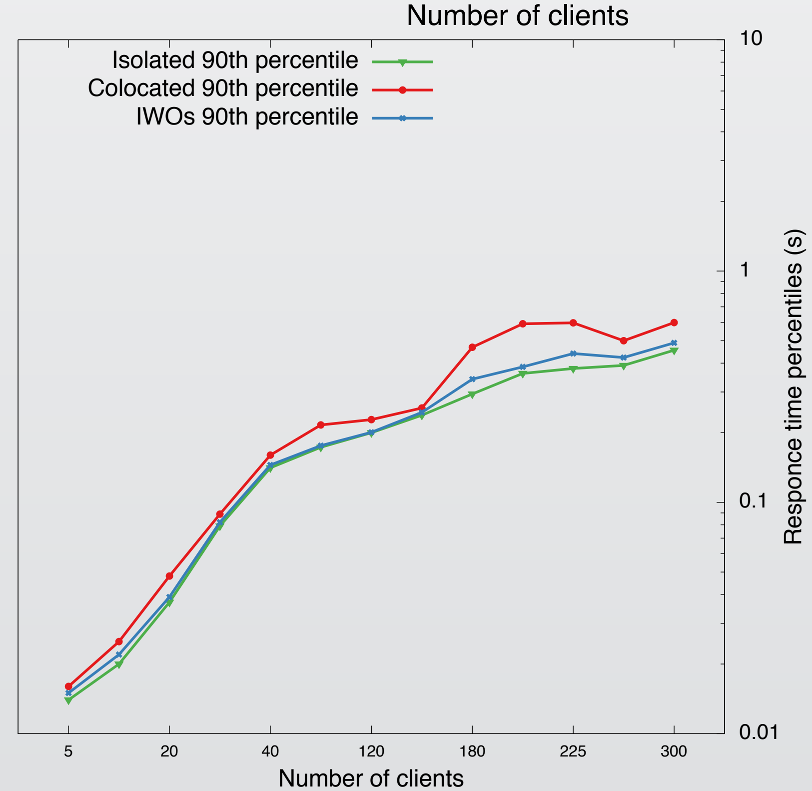
Systems Compared

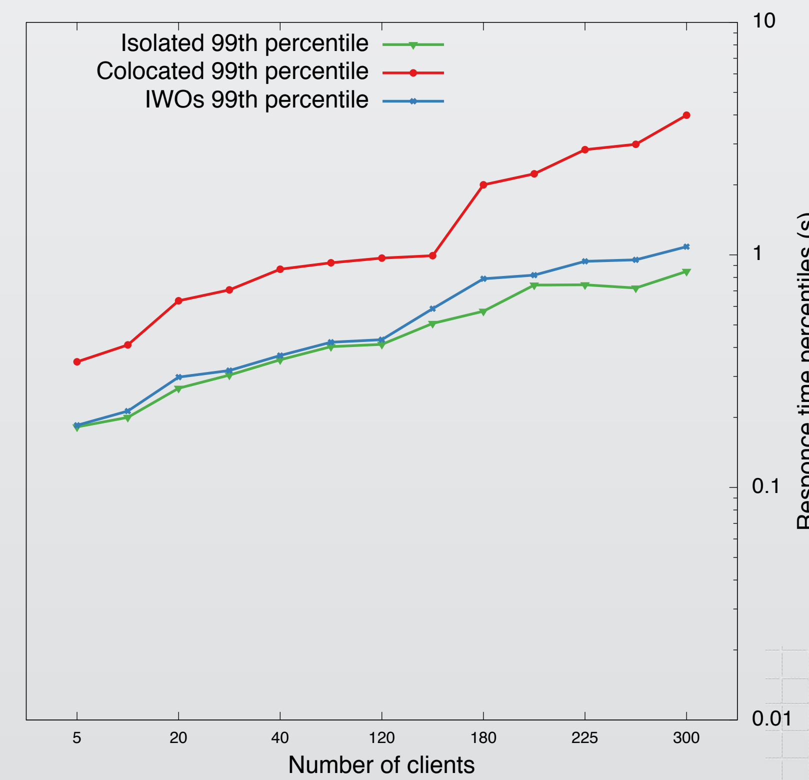
- **Isolated** Play2SDG
  - Play framework, Cassandra and Spark are configured to use up to 2 cores and 2GB of memory each through the Mesos API
  - Spark is set up in cluster mode and was **not allowed** to be colocated with Play application
- **Colocated** Play2SDG
  - Play framework, Cassandra and Spark are configured to use up to 2 cores and 2GB of memory each through the Mesos API
  - Spark is set up in cluster mode and was **allowed** to be colocated with Play application
- Play2SDG **IWOs** implementation
  - both serving and analytics tasks implemented as an SDG
  - configure the application JVM to use the same resources as above using JVM configuration and cgroups - disabled scheduling

Play2SDG Case Study Results



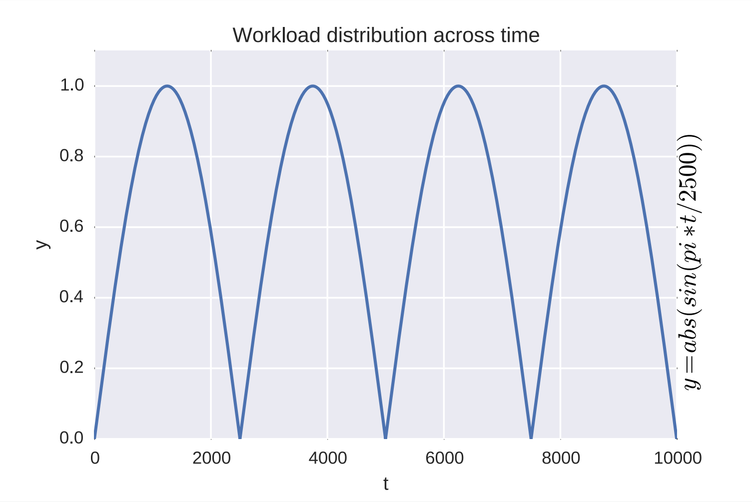


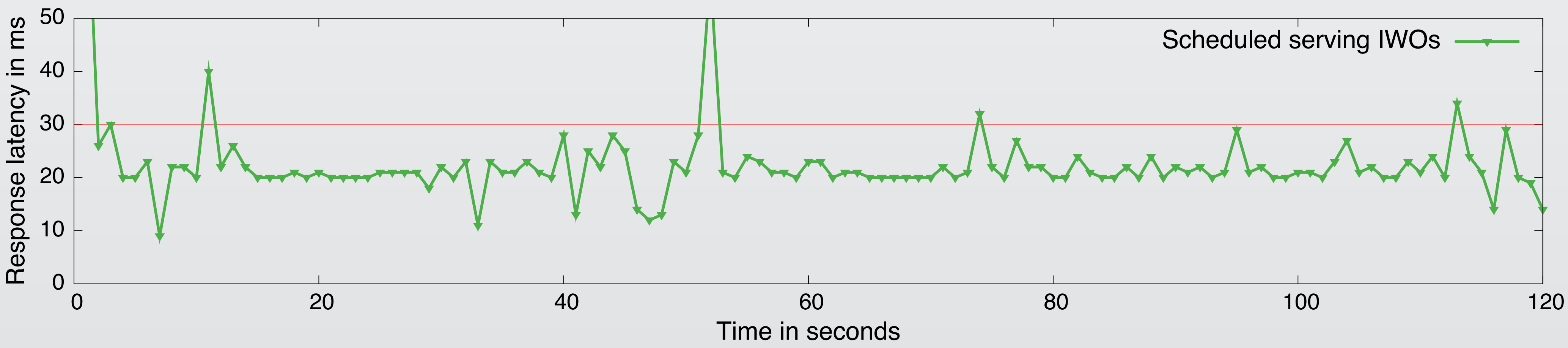




play2sdg

Scheduling Results





Thesis Contributions

- Introduced **In-memory Web Objects** (IWOs), offering a unified model to developers when writing web applications that have the ability to serve data while using big data analytics
- IWOs **isolation** mechanism that is based on cooperative task scheduling. Co-operative task scheduling reduces the scheduling decisions and allocates resources in a fine-grained way, leading to improved resource utilisation
- The evaluation of IWOs by implementing **Play2SDG,** a real web application similar to Spotify, with both online/LC and offline/BE tasks. The web application was implemented as an extension of Play framework

Future work

- Focus on efficient **distributed scheduling** of BE analytics and LC serving tasks
- Further investigate the **automatic conversion** of a web application in an SDG
- Implement IWOs abstract programming model for other stateful stream processing frameworks like Flink
- More Evaluation!

Questions???

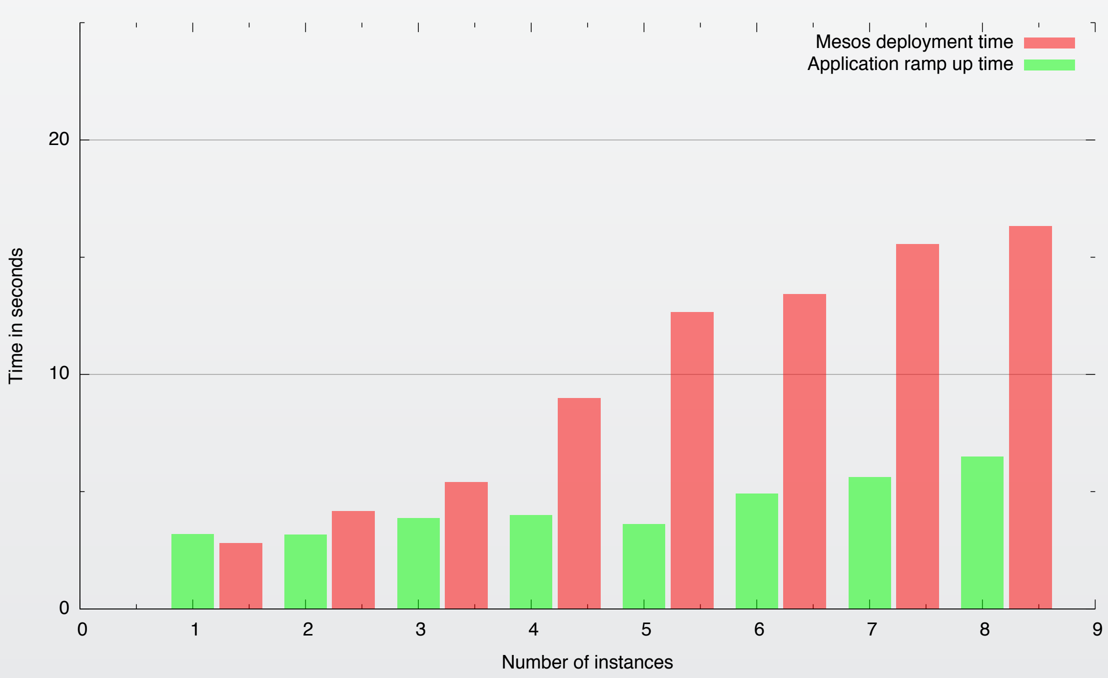
Thank you!

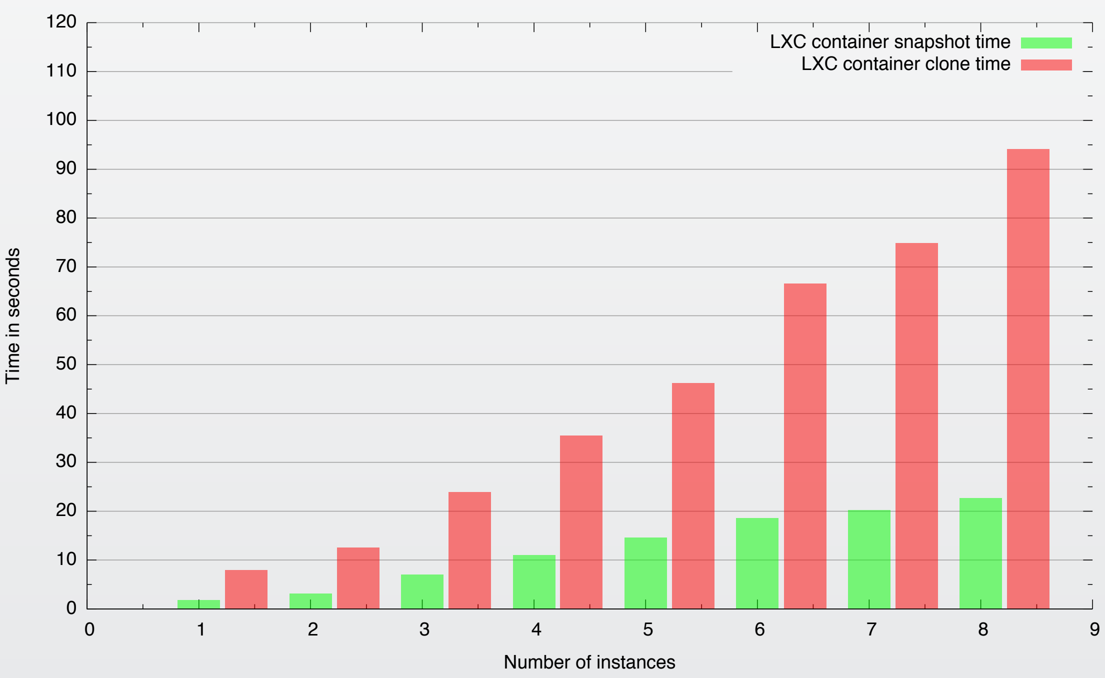
email: pgaref@imperial.ac.uk

Demo time!

Imperial College London

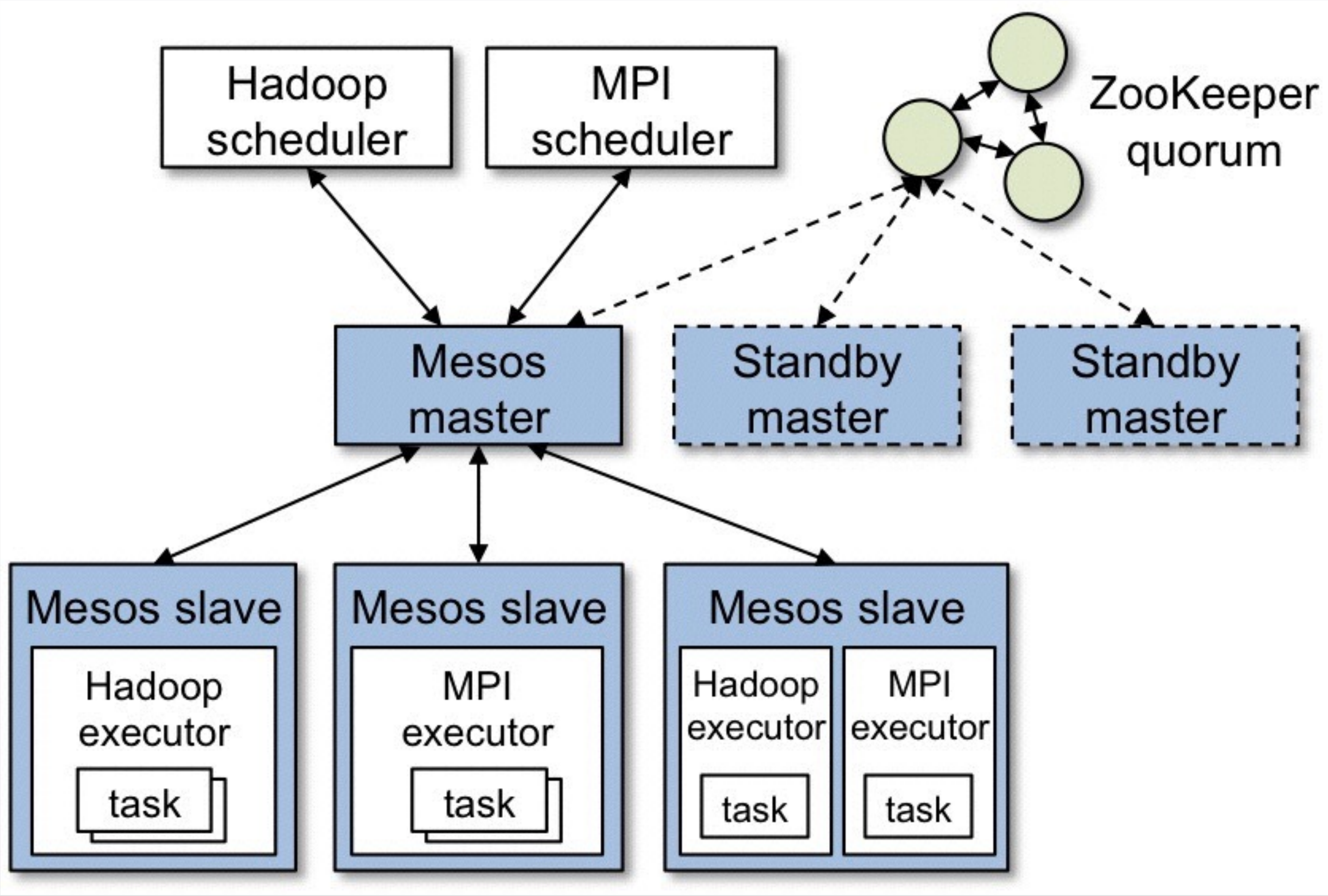
Backup Slide Isolation





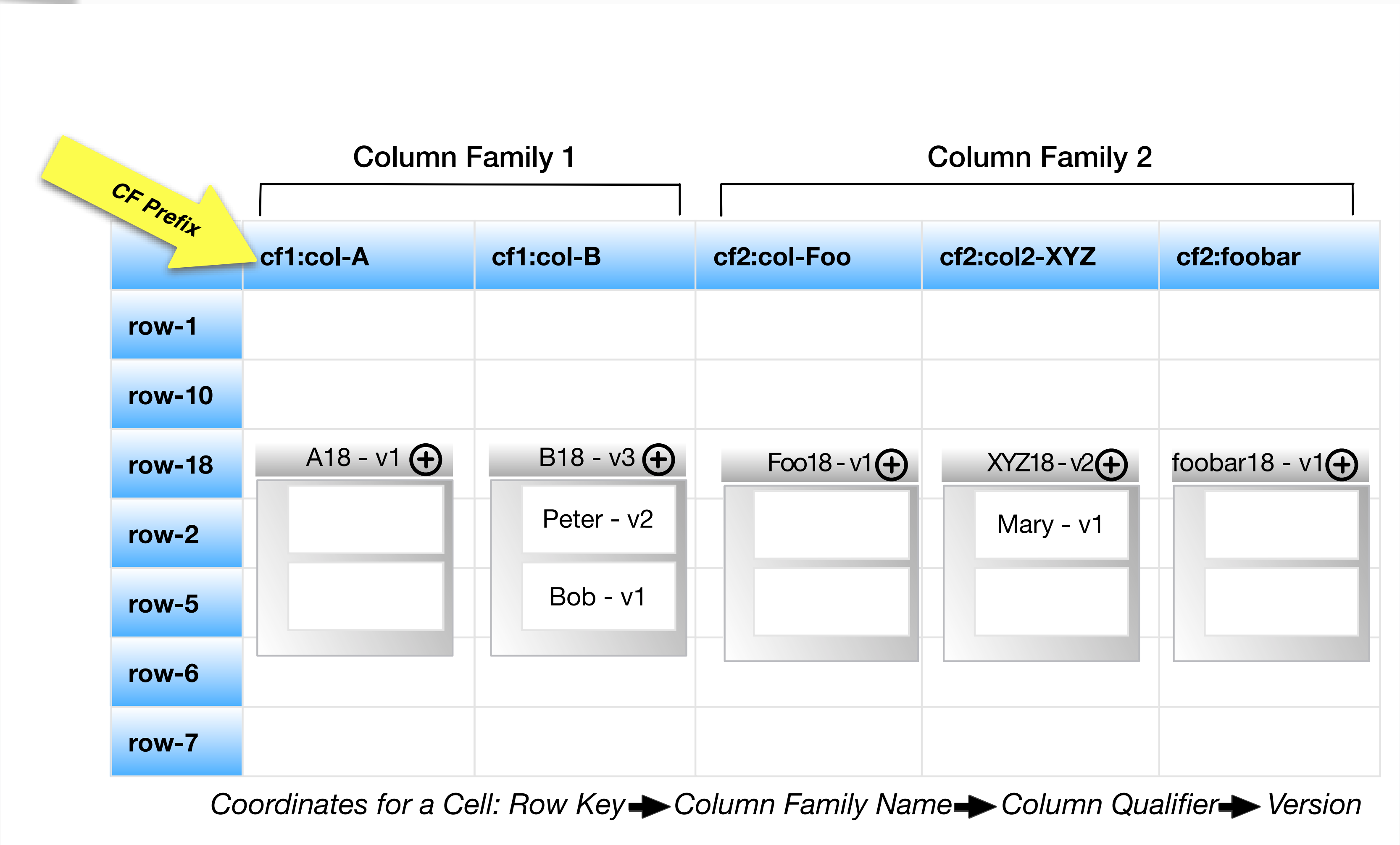
play2sdg

Backup Slide 2



Imperial College London

Backup Slide 3



Row Key	Cluster Key	Column Family		Cluster Key	Column Family	
id	**TimeSt.**	**StatsMap**	**Description**	**TimeSt.**	**StatsMap**	**Description**
sparkCF	timeX	Map<k,v>	Stat Desc.	timeX	Map<k,v>	Stat Desc.
playServ	timeY	Map<k,v>	Stat Desc.	timeY	Map<k,v>	Stat Desc.
playCF	TitleZ	Map<k,v>	Stat Desc.	TitleZ	Map<k,v>	Stat Desc.

Row Key	Static Column Families			Dynamic Column Family	
id	**Title**	**Artist**	**releaseDate**	**Tag**	**Tag**
0axfdsg	TitleX	name	DateTime	tag1	…
0axfdsa	TitleY	name	DateTime	tag2	…
0axfdsb	TitleZ	name	DateTime	tag1	…

play2sdg