How In-Memory solutions assist with SaaS integrations

Craig Gresbrink – Solutions Architect – 24 Hour Fitness

In-Memory Computing | SUMMIT

24 HOUR FITNESS

Intro

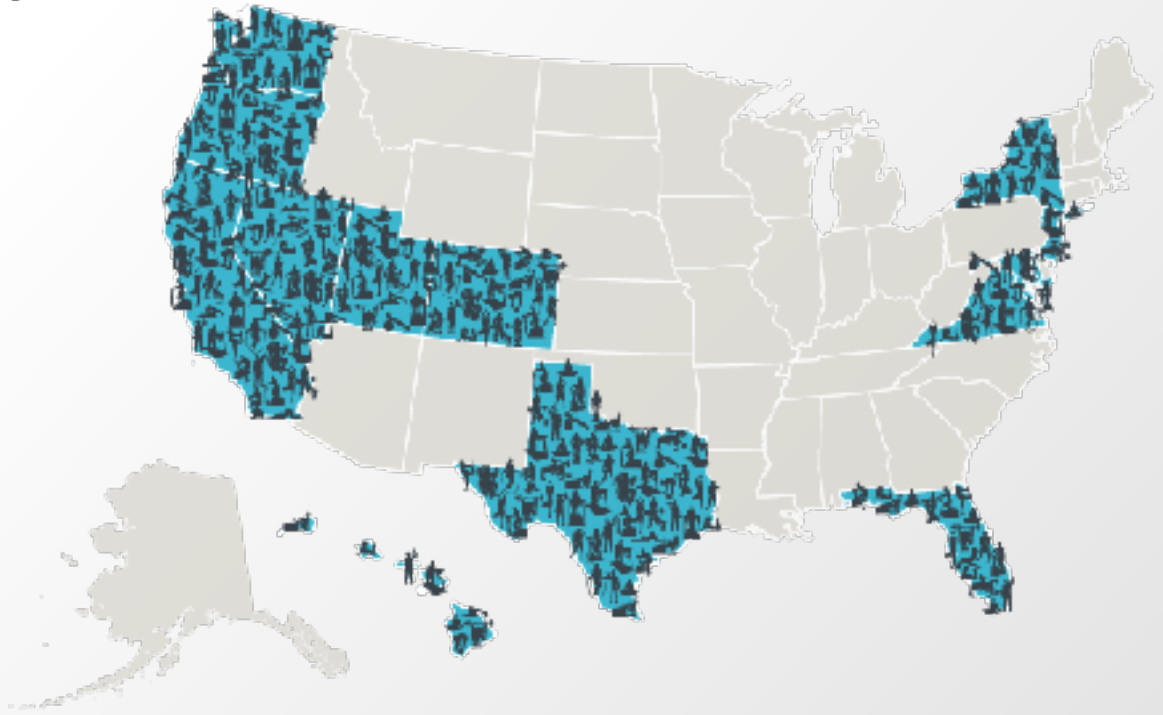
Craig Gresbrink
cgresbrink@24hourfit.com
Solutions Architect

YOUR CODE DOESN'T CONFORM TO MY ARCHITECTURAL GUIDELINES.
THAT'S BECAUSE YOU'RE BAD AT YOUR JOB AND I'M GOOD AT MINE.
I DON'T KNOW HOW TO RESPOND TO THAT.
MAYBE YOU COULD ASK SOME-ONE WHO KNOWS HOW TO DO YOUR JOB.
Dilbert.com @ScottAdamsSays
7-22-17 © 2017 Scott Adams, Inc./Dist. by Andrews McMeel

Intro – 24 Hour Fitness

24 Hour Fitness - 400+ clubs in 13 States

- We are a leading fitness industry pioneer with nearly four million members in more than 400 clubs across the U.S. For more than 30 years, we've held fast to our mission of helping people improve their lives through fitness.



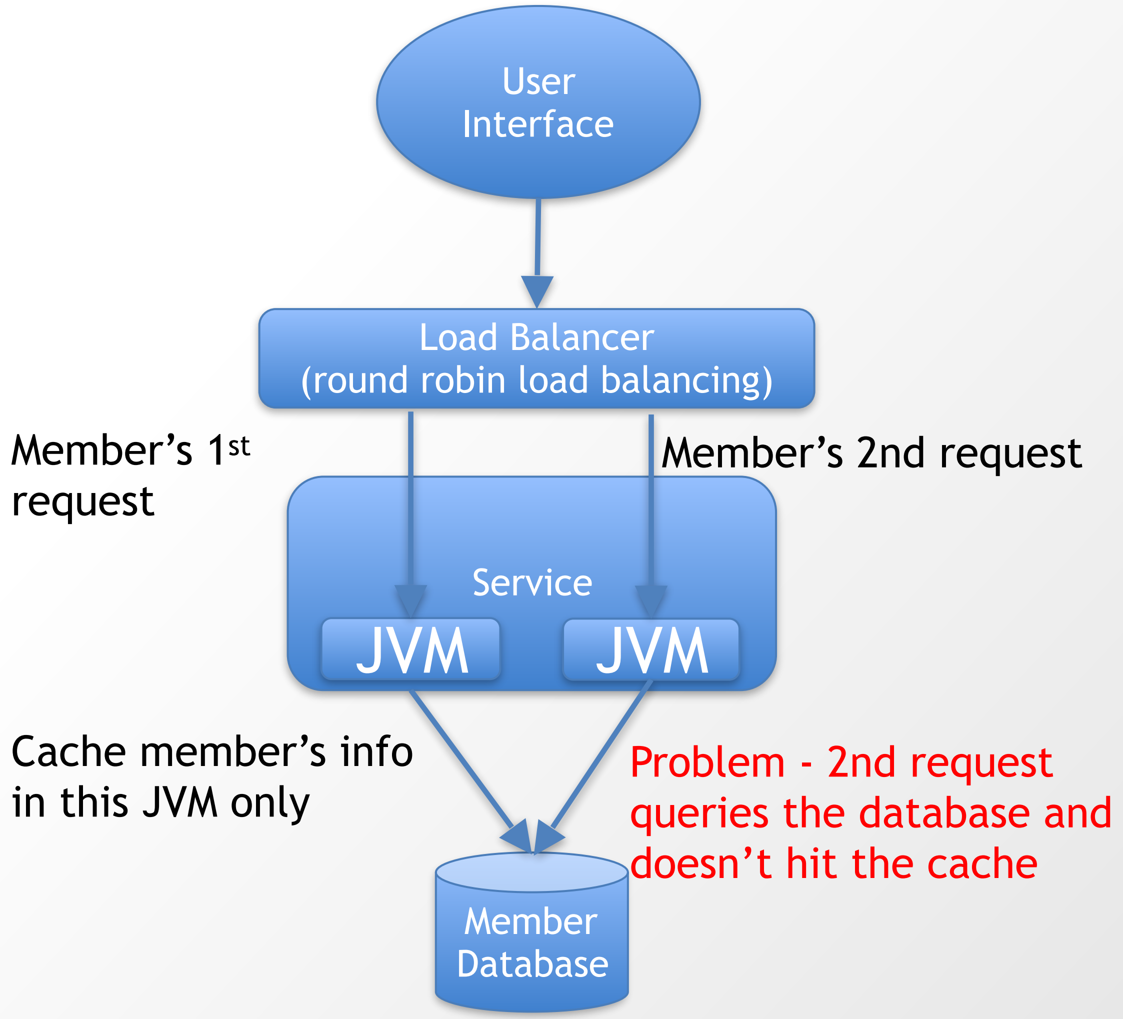
What am I going to show you?

How In-Memory solutions assist with SaaS integrations

Agenda

- 24 Hour Fitness' historical architecture and some limitations
- How in-memory solutions solved use cases at 24 Hour Fitness
- Issues we ran into
- How in-memory solutions might be leveraged in the future at 24 Hour Fitness
- Q &A
- Reception!

24 Hour Fitness - Historical architecture and caching

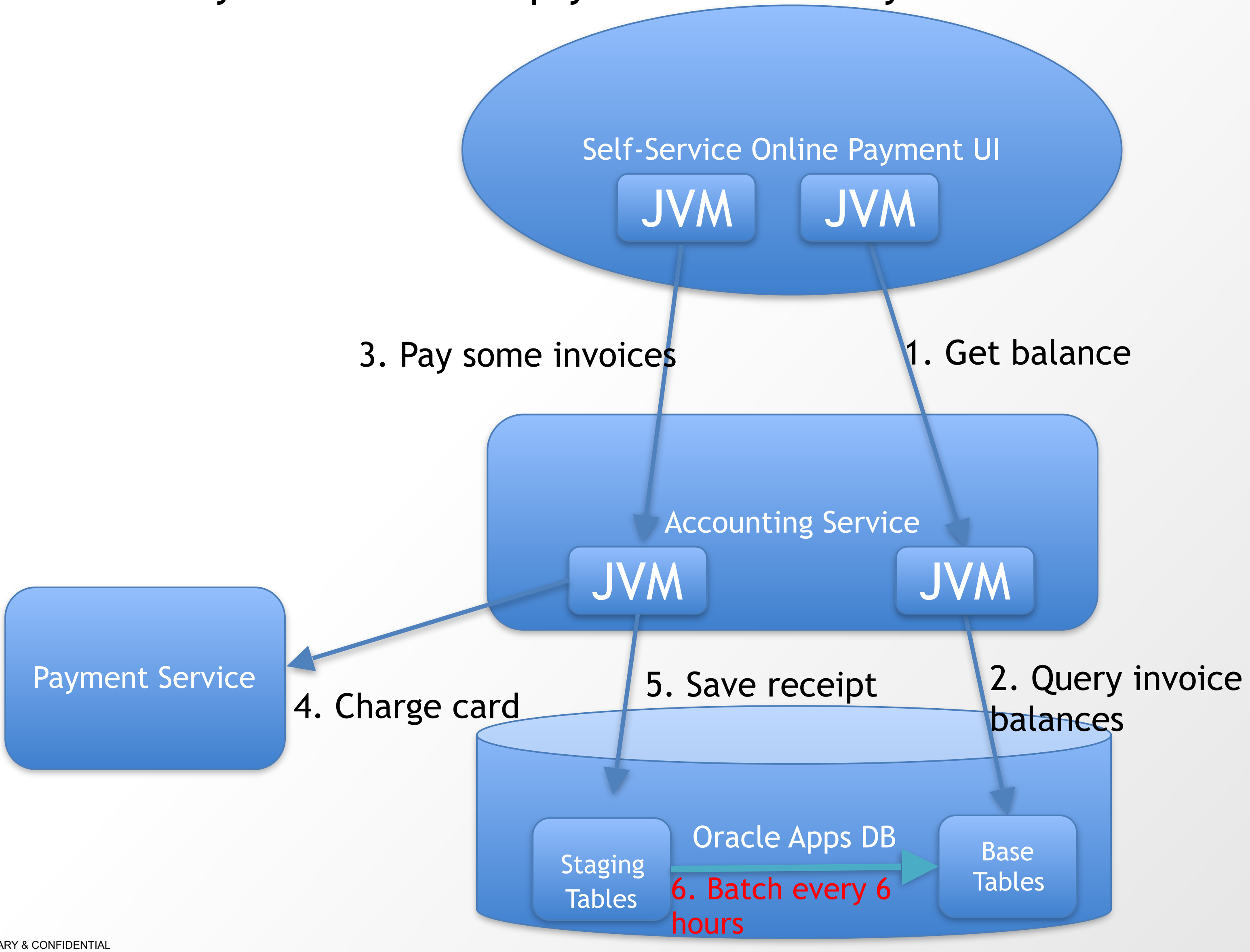

User Interface
Load Balancer (round robin load balancing)
Member's 1st request
Member's 2nd request
Service
JVM
JVM
Cache member's info in this JVM only
Problem - 2nd request queries the database and doesn't hit the cache
Member Database

We cache in each JVM only. In our setup there is no guarantee which JVM a user will hit due to round robin load balancing (no sticky session).

More cache misses as we scale (JVMs) horizontally.

Historical architecture and caching

Use Case 1 - Why it won’t work for payments to a batch system



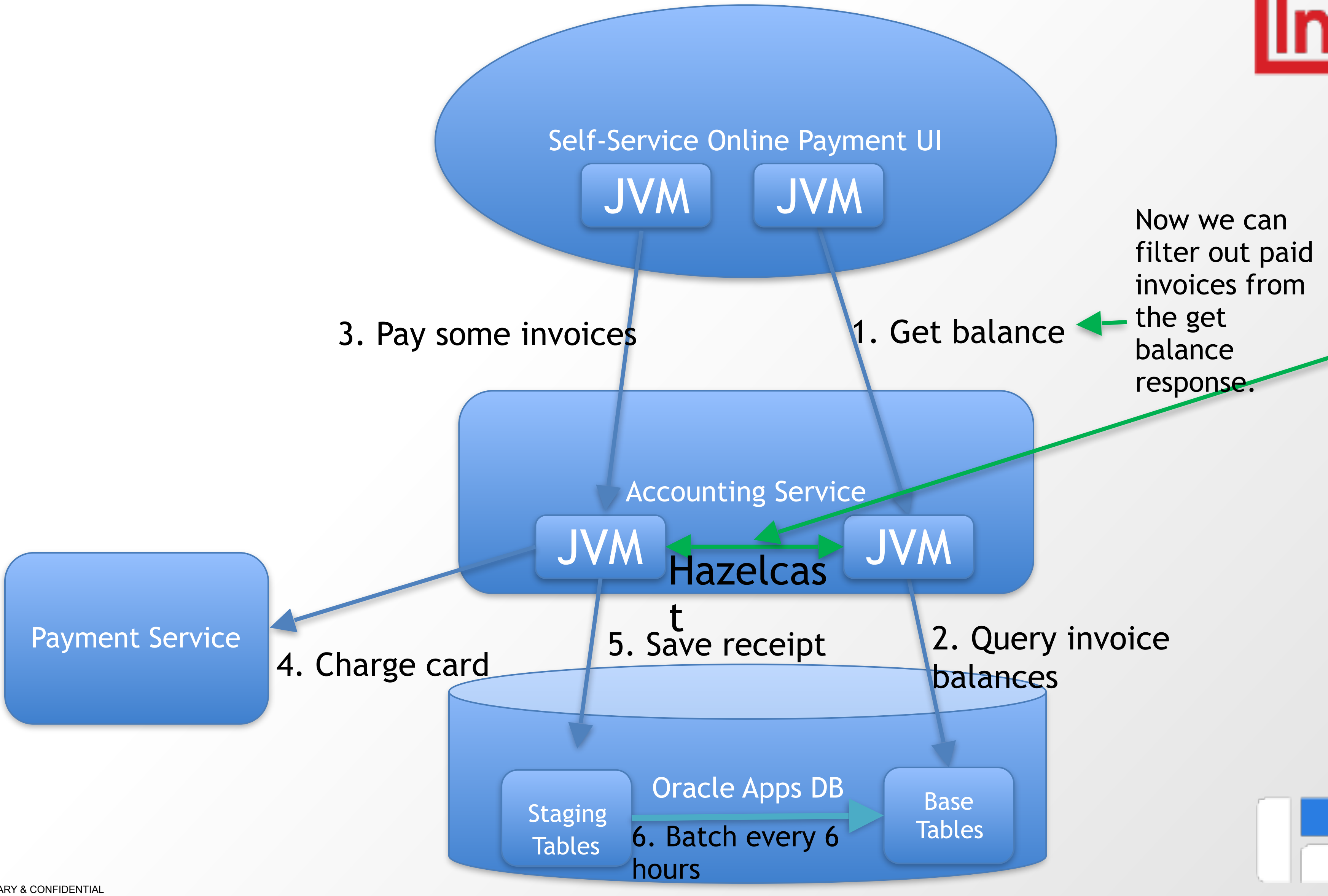
Members, can pay their bill(s) online via our self-service customer portal.

What happens when they come back 2 hours after making their payment?

We could’ve solved it by querying the staging tables as well as the base tables. It would be slow for all customers. 99% haven’t made a previous

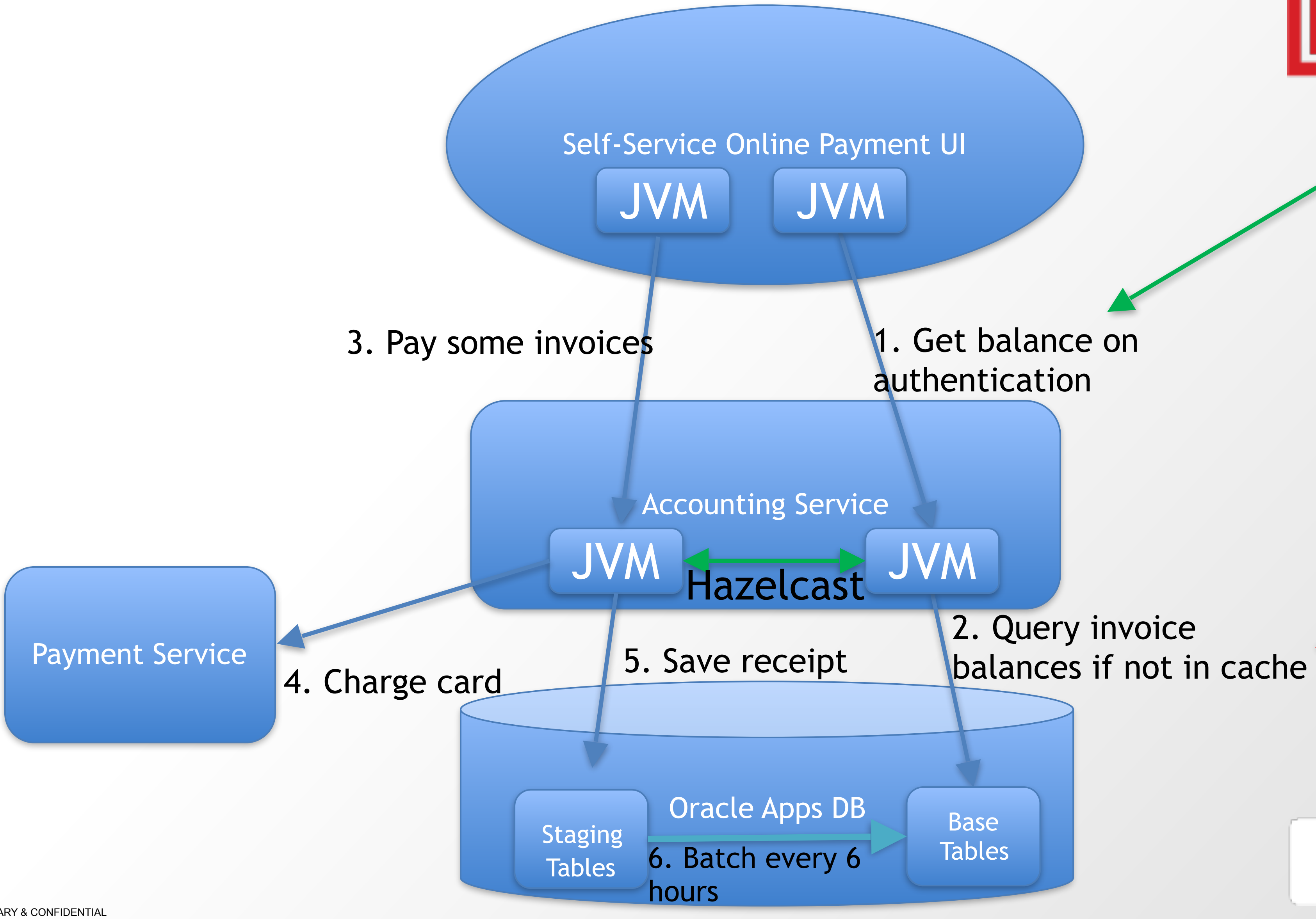
A distributed cache solves our problem…

Hazelcast to the rescue for use case 1.



What if we implement a distributed cache such that cache consistency is retained across the JVMs so we know which invoices have already been paid?

A distributed cache solves another problem

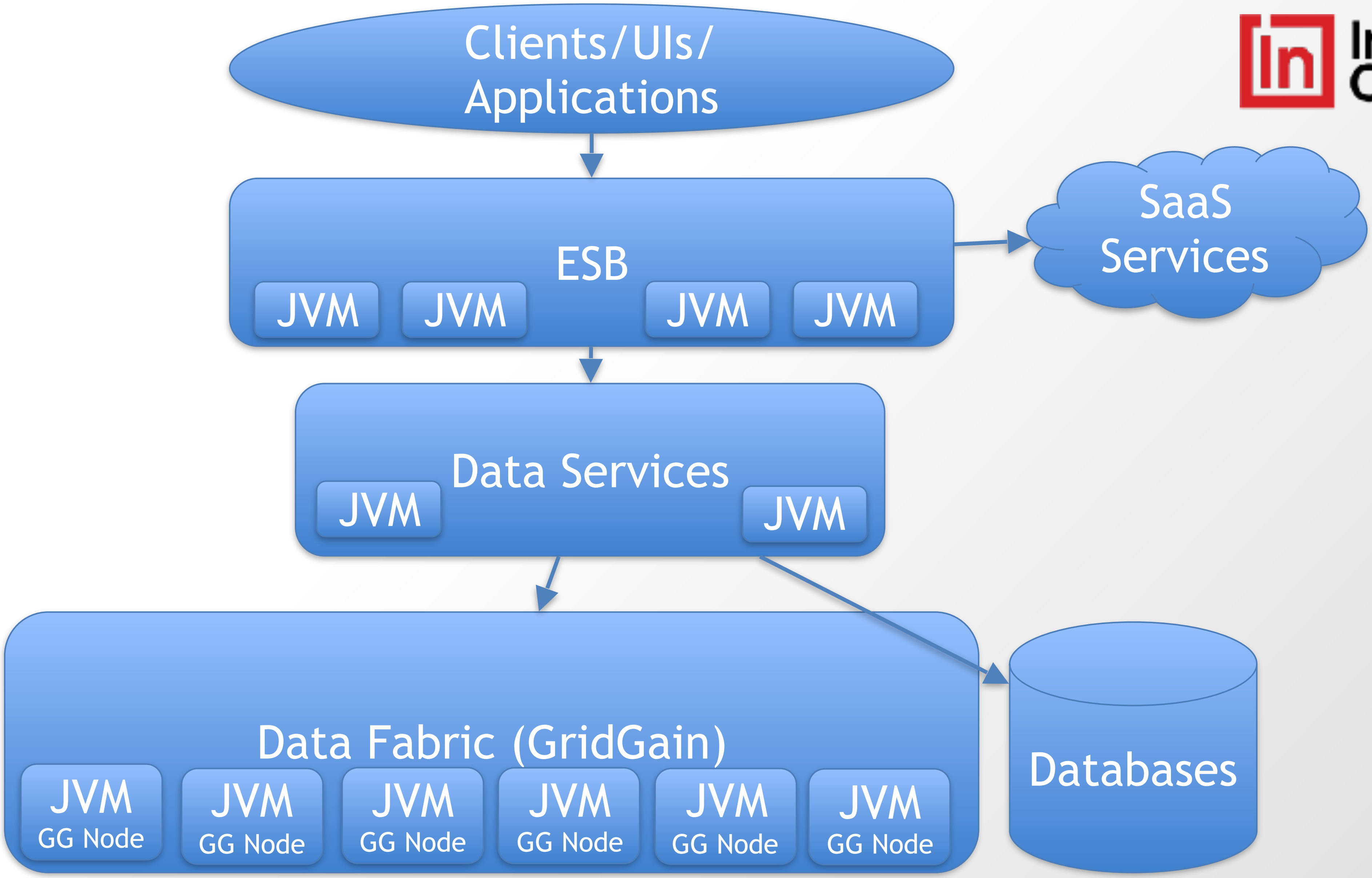


Eagerly cache, or pre-cache, invoice balances.

Don't wait 'til they go to the make a payment screen. A better Customer Experience (CX).

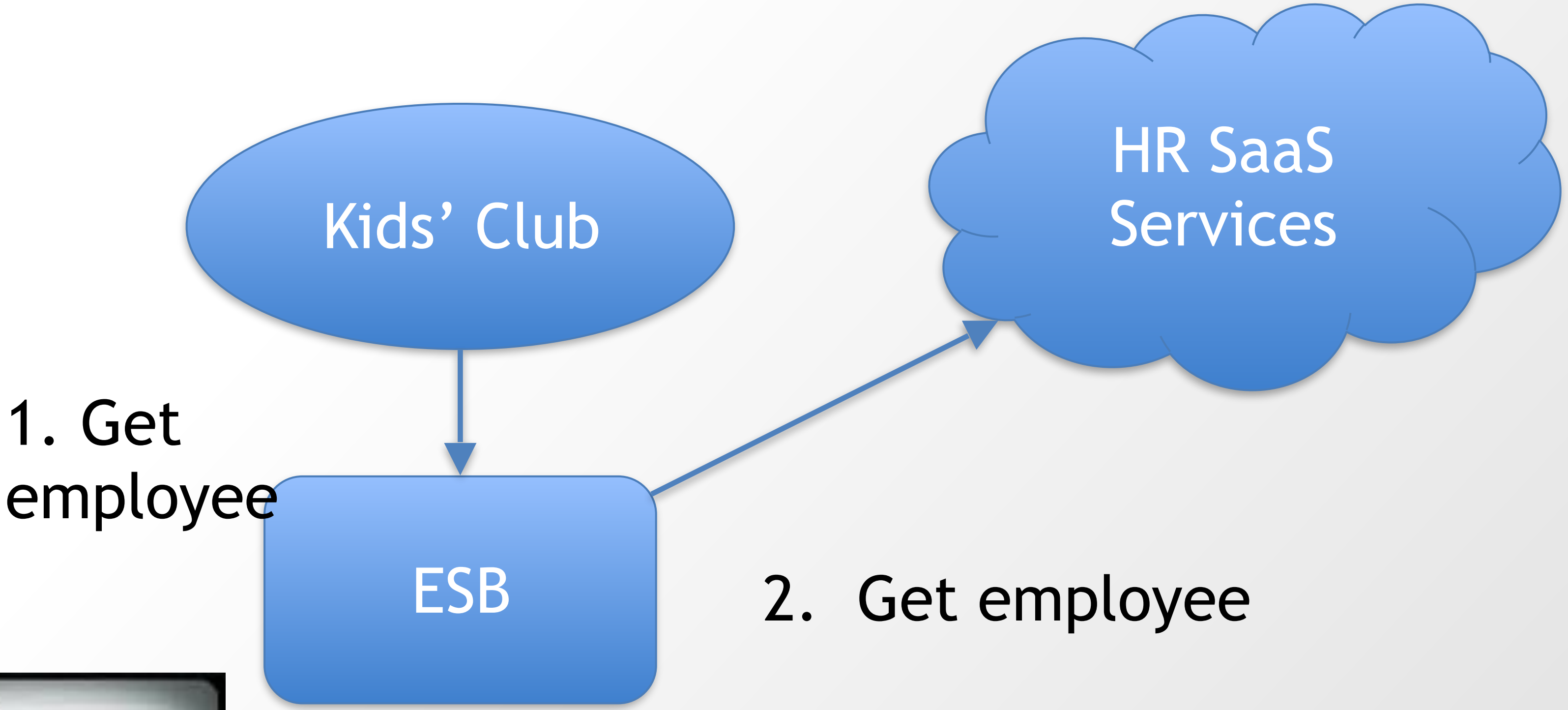
But there is another problem, for some customers this query takes 40 seconds.

Current architecture and caching

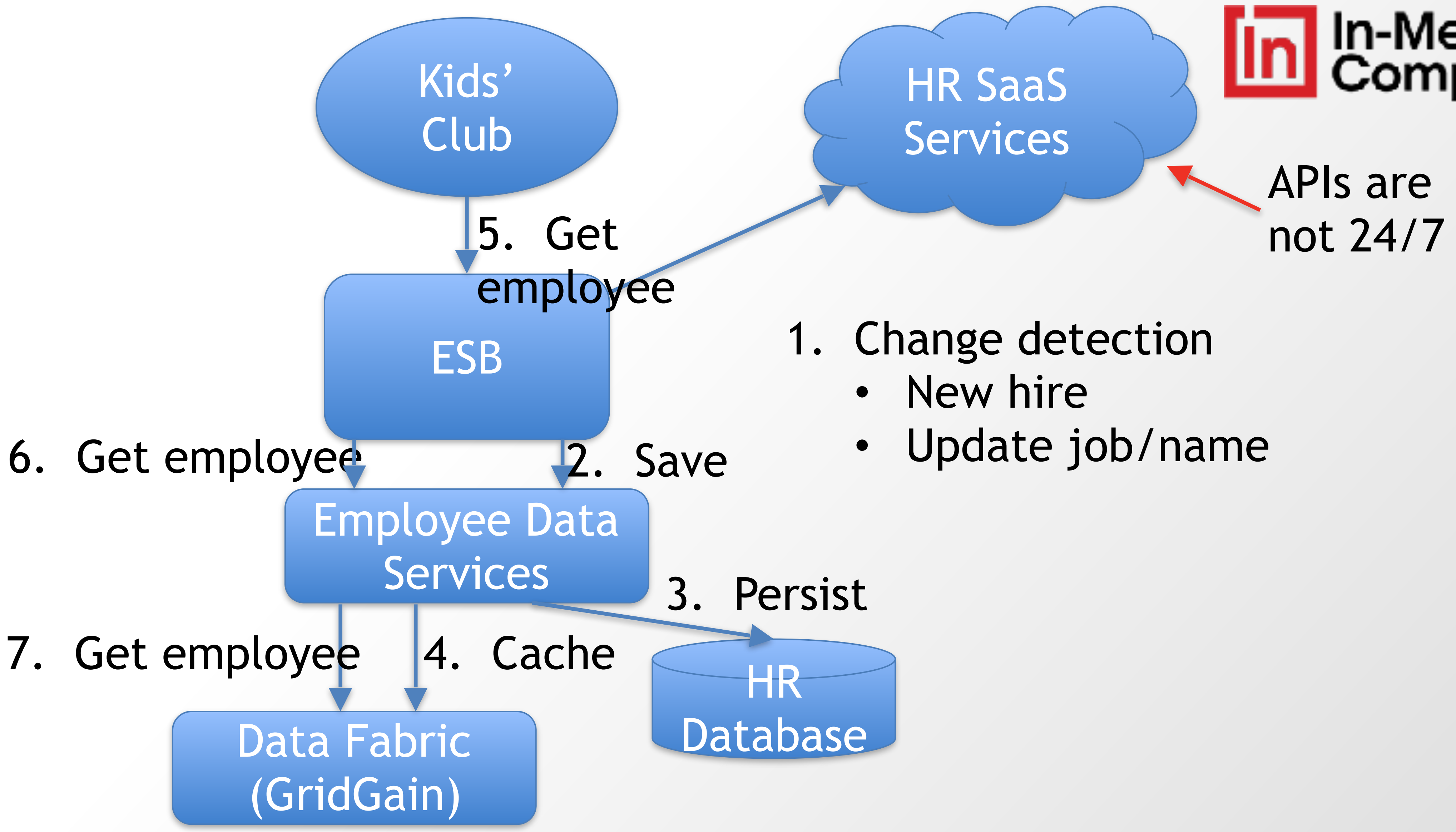


Nirvana with HR SaaS system

Life is perfect!



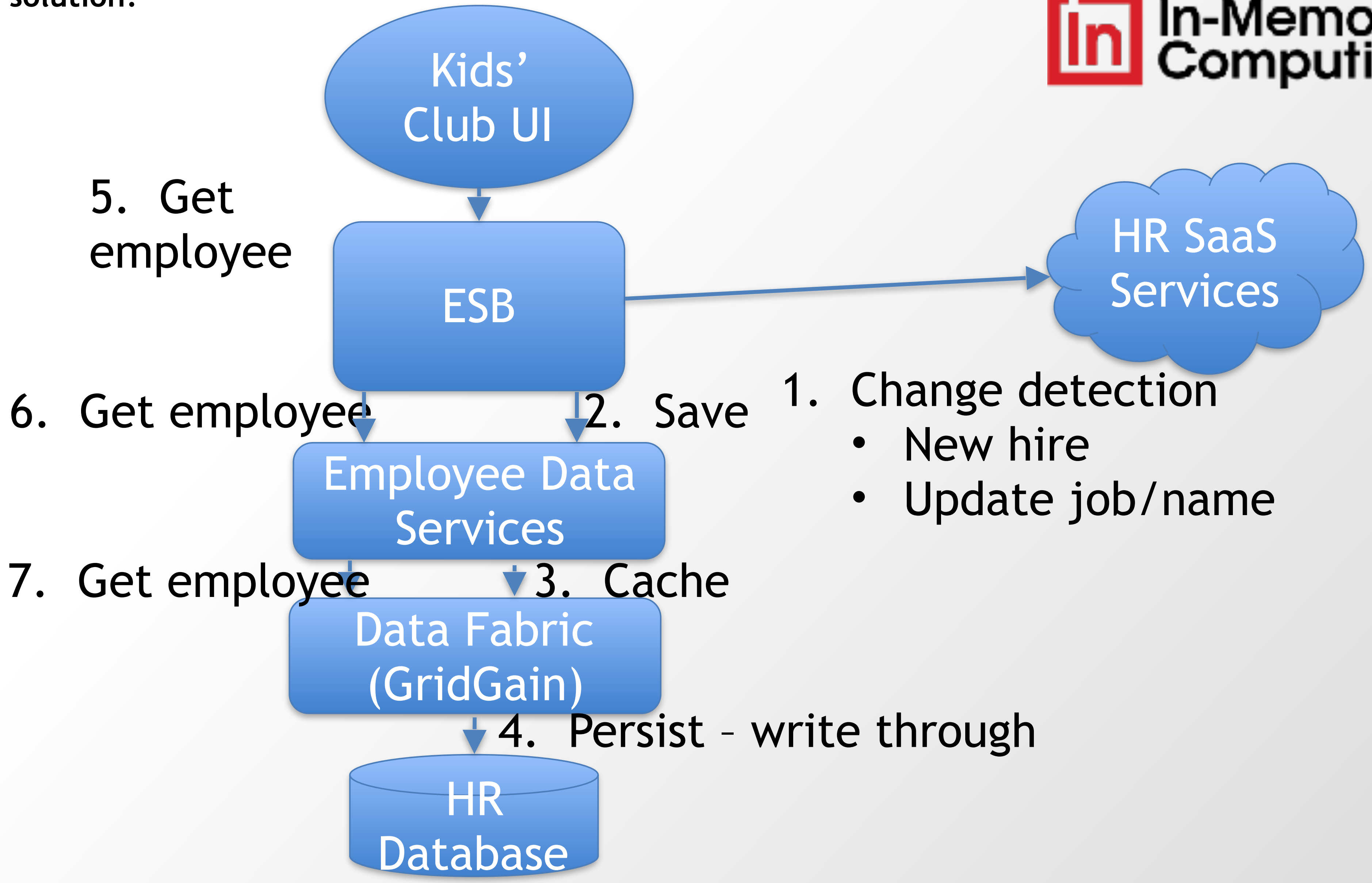
Use case 2: Vendor's APIs are not 24/7, but our business is 24/7



1. Change detection
   - New hire
   - Update job/name

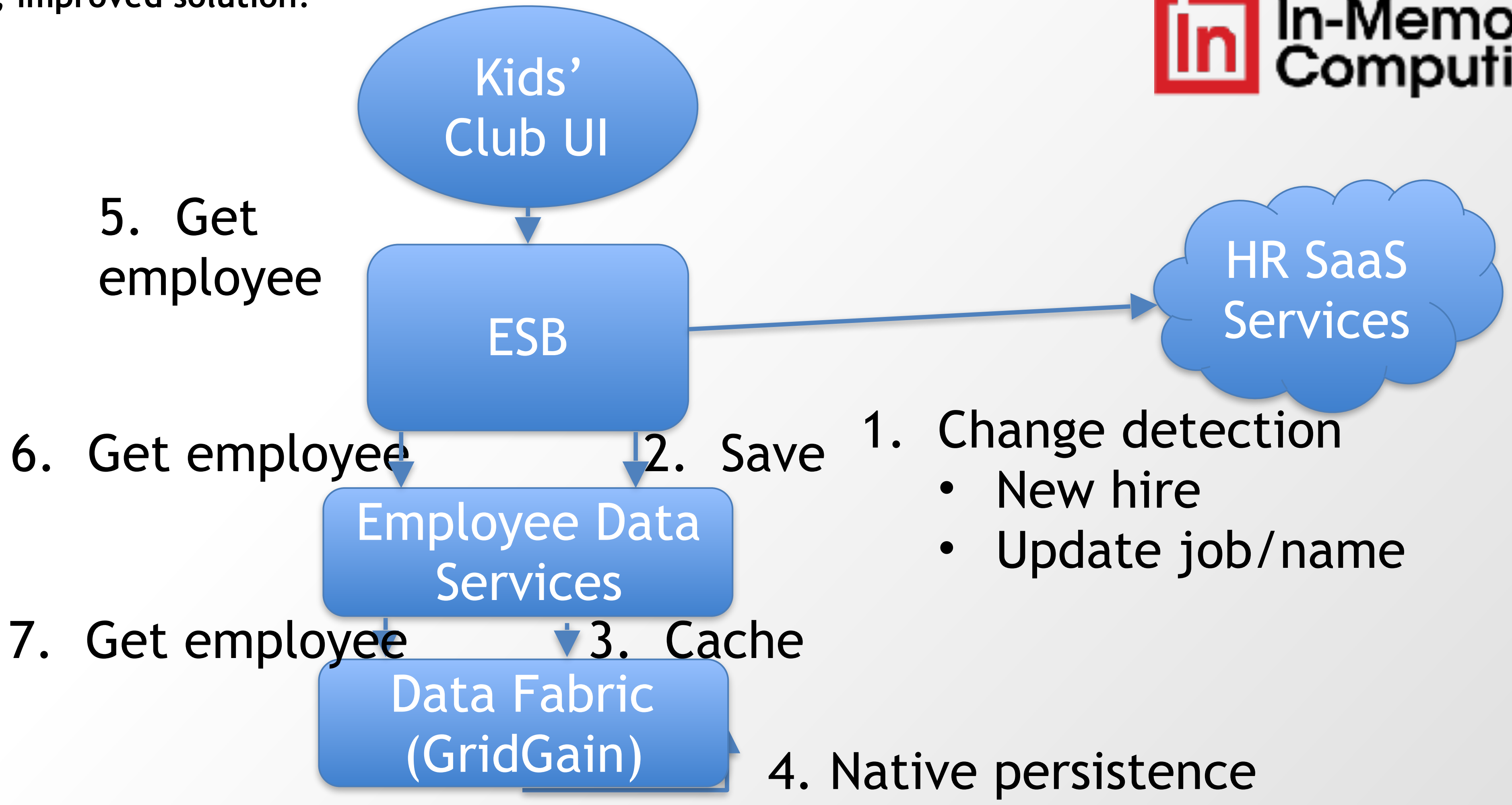
Use case 2: Vendor's APIs are not 24/7, but our business is 24/7

An improved solution?



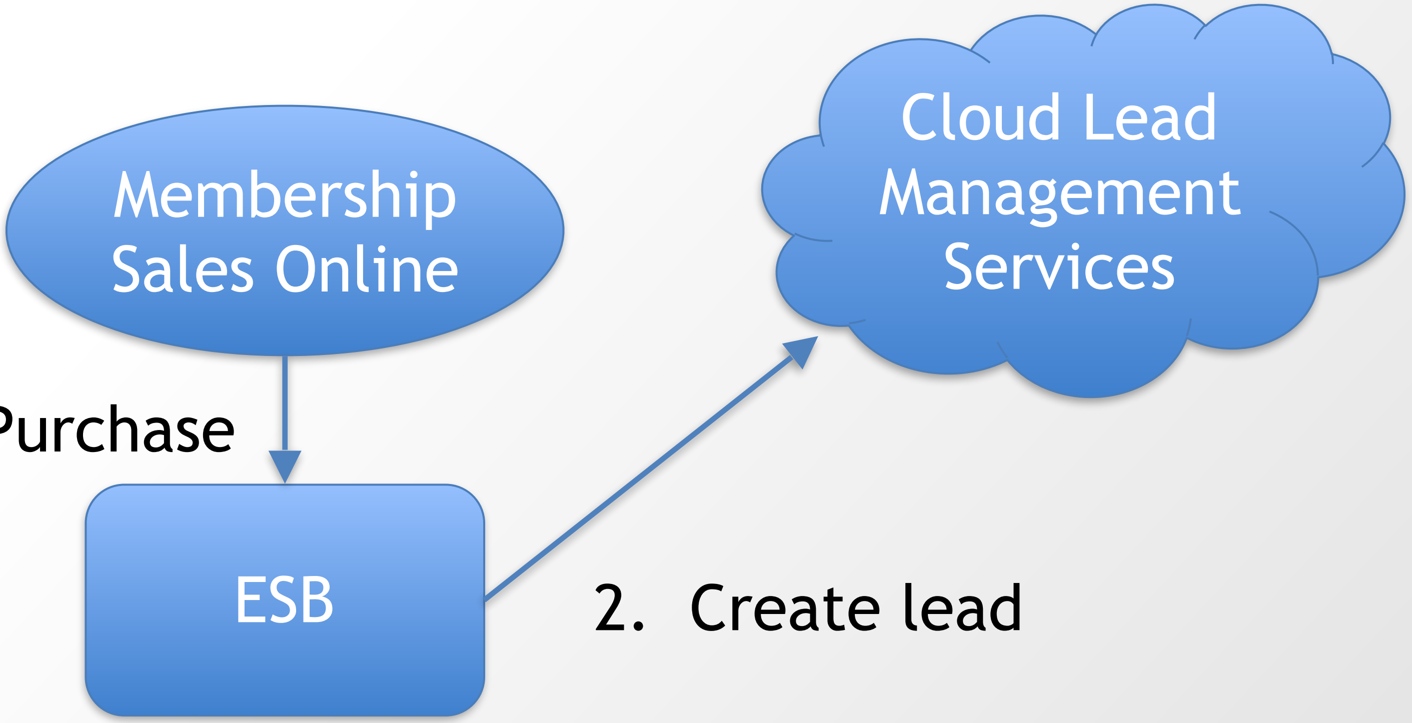
Use case 2: Vendor's APIs are not 24/7, but our business is 24/7

An improved, improved solution?



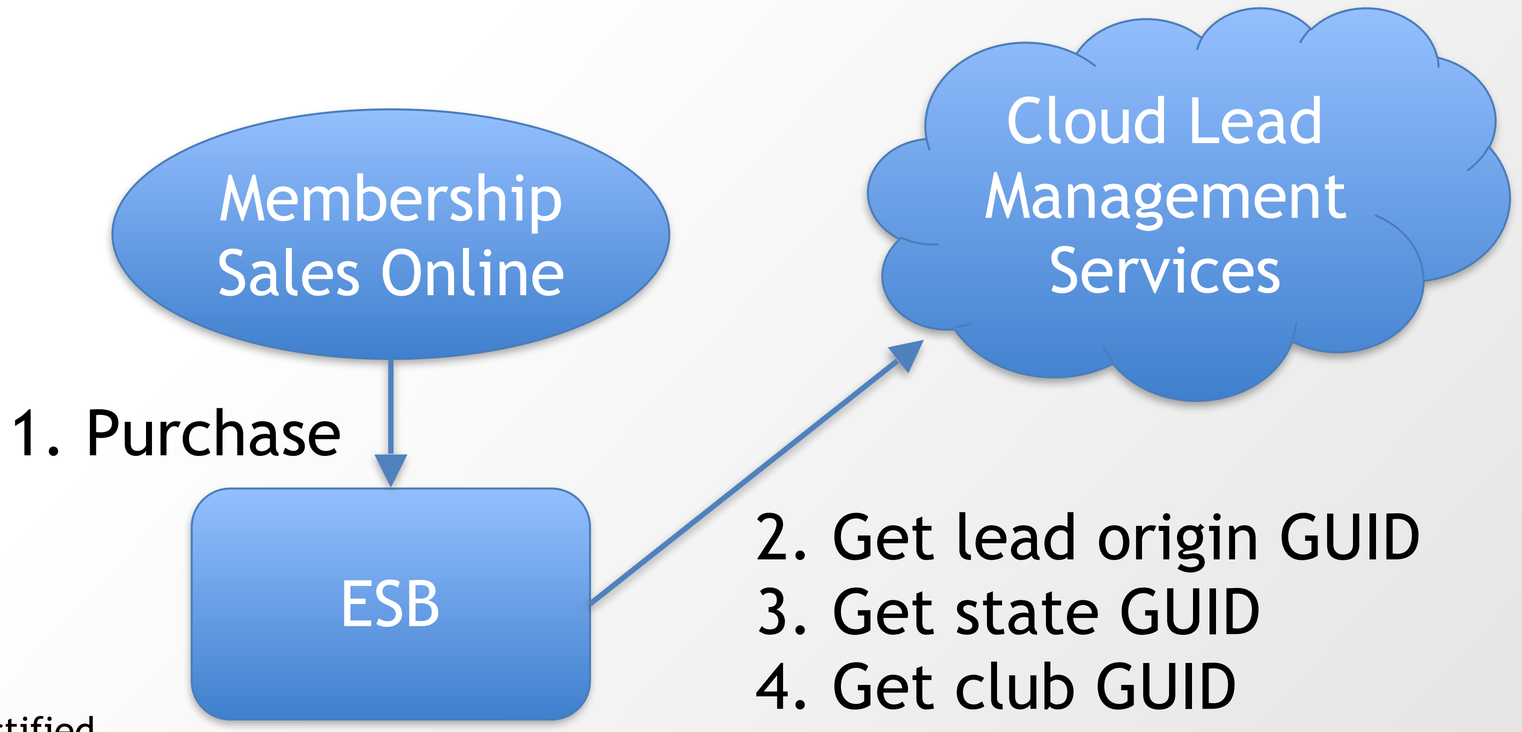
Use case 3: A chatty real-time integration with CRUD-based APIs is too slow

Nirvana, finally!



Use case 3: A chatty real-time integration with CRUD-based APIs is too slow

It's not that easy! We have to make multiple calls to LMS to create a lead…

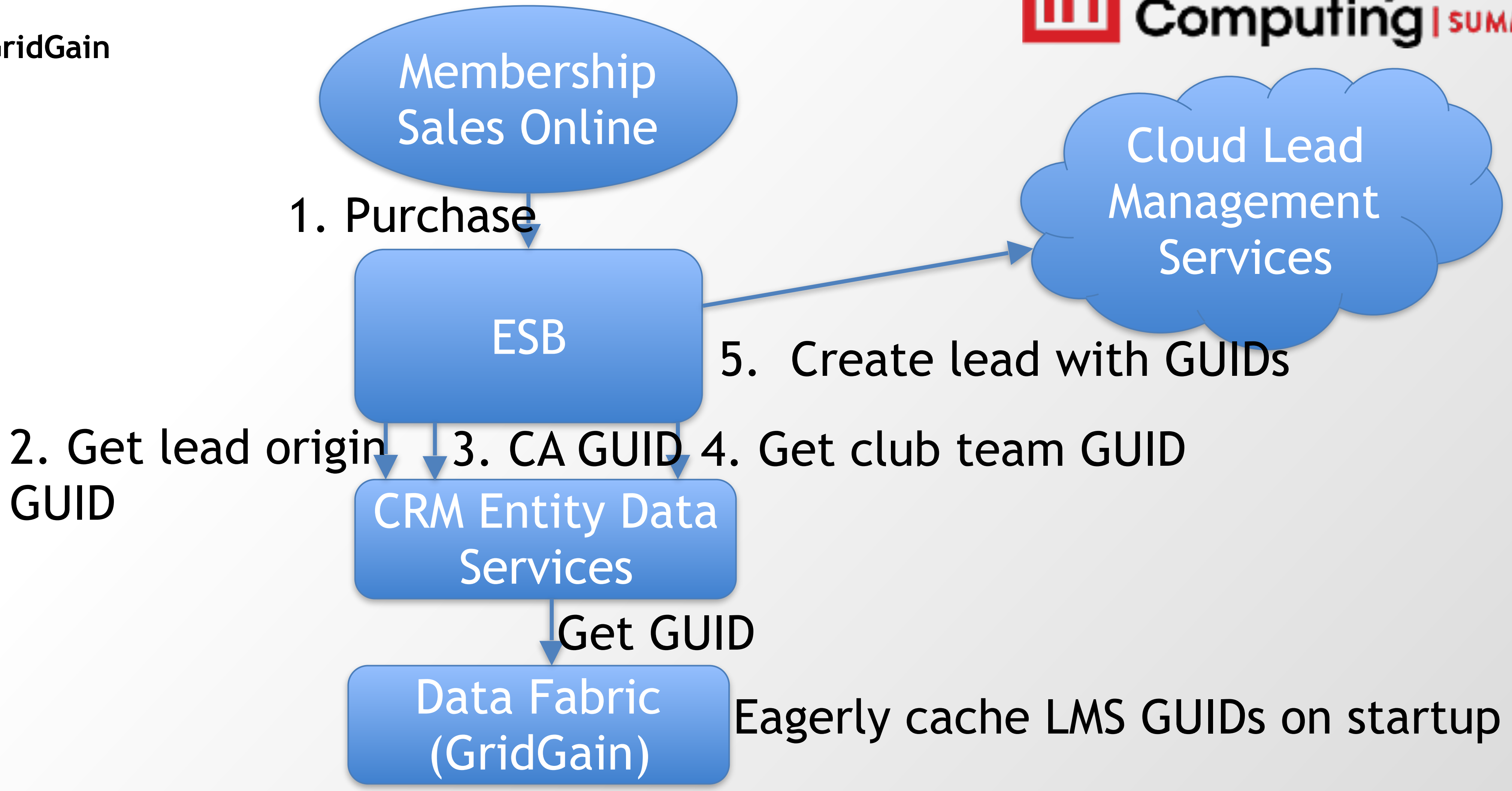


2. Get lead origin GUID
3. Get state GUID
4. Get club GUID
5. Create lead with GUIDs

Disclaimer:
Chatty APIs have been rectified in future versions of the product.

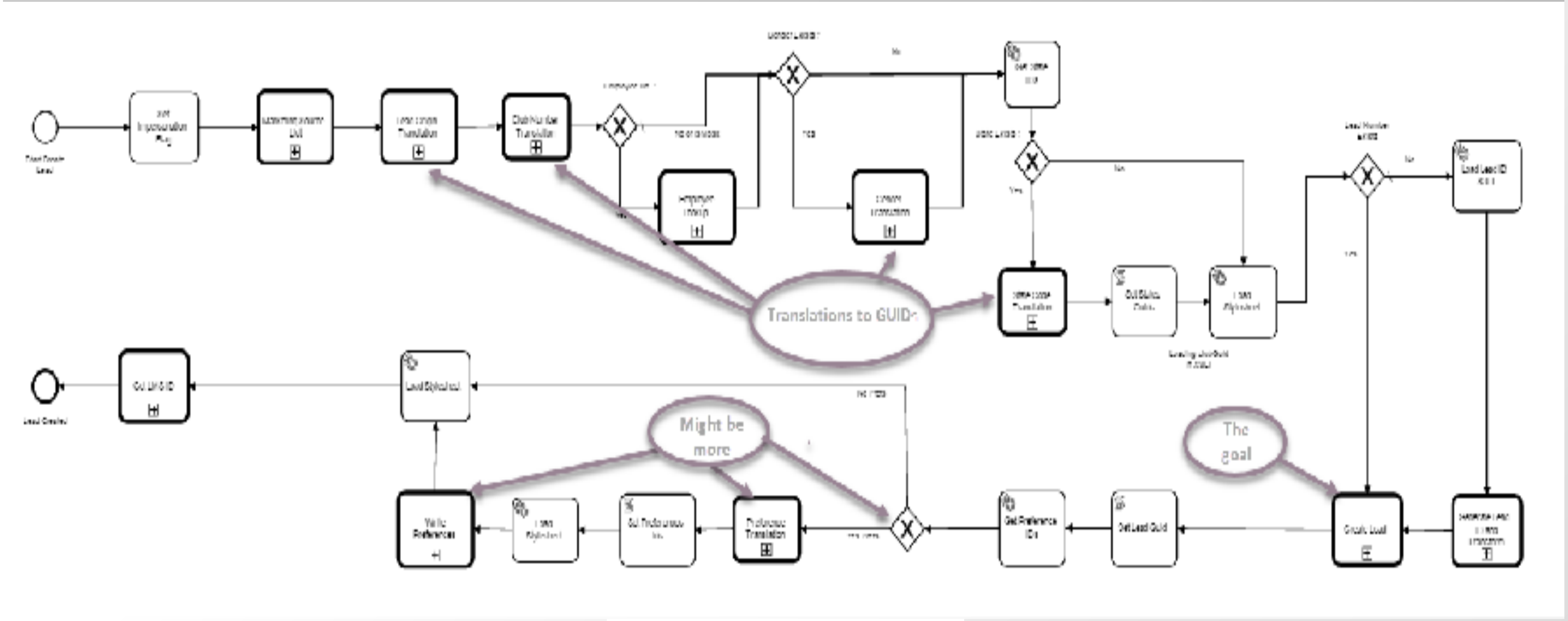
Use case 3: A chatty real-time integration with CRUD-based APIs is too slow

Data flow with GridGain



Use case 3: A chatty real-time integration with CRUD-based APIs is too slow

Lead Management System and GUID translations in Camunda BPMN

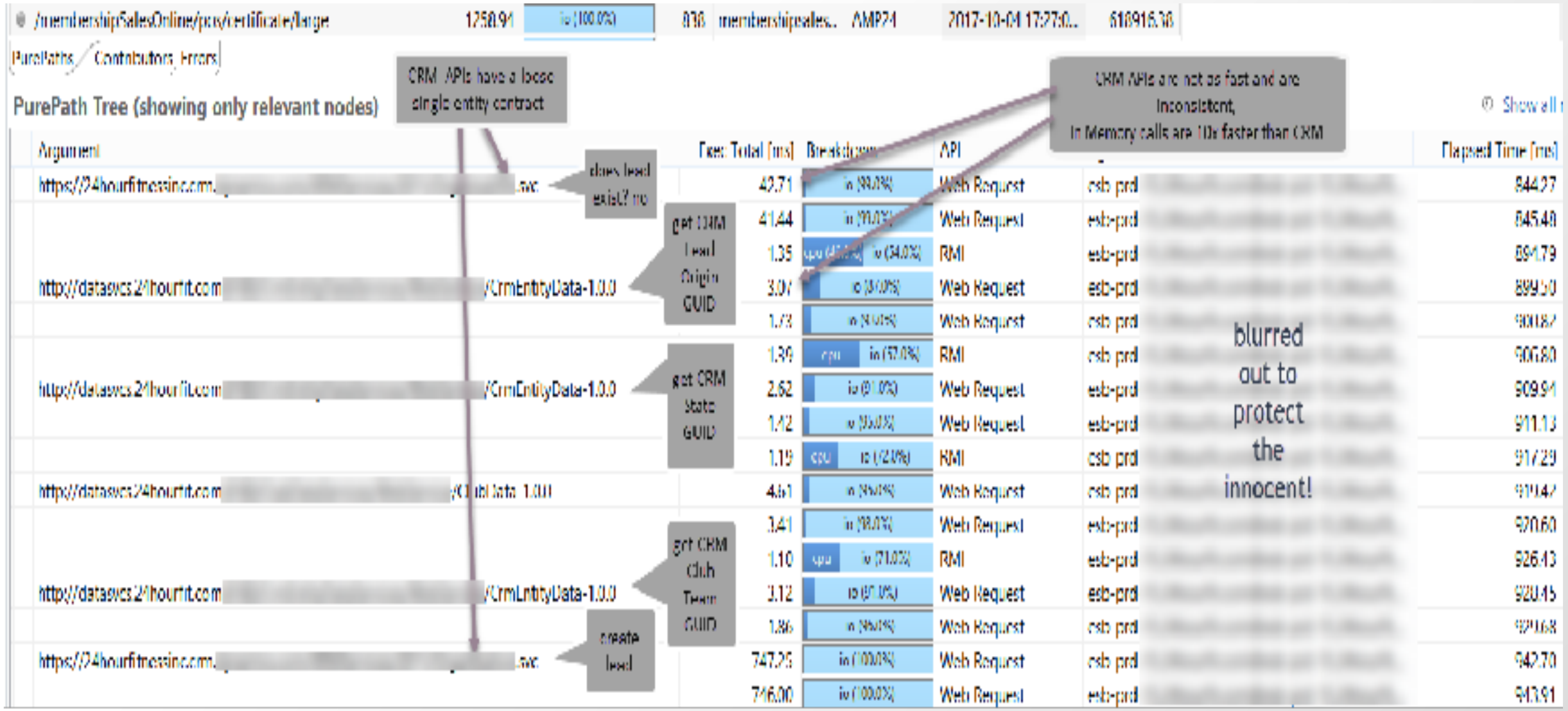


Camunda

PROPRIETARY & CONFIDENTIAL

Use case 3: A chatty real-time integration with CRUD-based APIs is too slow

Lead Management System and GUIDs translations – How fast is the cache?


CRM APIs have a loose single entity contract
CRM APIs are not as fast and are Inconsistent,
In Memory calls are 10x faster than CRM
does lead exist? no
get CRM Lead Origin GUID
get CRM State GUID
get CRM Club Team GUID
create lead
blurred out to protect the innocent!

/membershipSalesOnline/pos/certificate/large 1250.94 io (100.0%) 838 membershipsales... AMP24 2017-10-04 17:27:0... 618916.38

PurePaths / Contributors Errors

PurePath Tree (showing only relevant nodes)

Show all

Argument	Exec Total [ms]	Breakdown	API		Elapsed Time [ms]
https://24hourfitnessinc.com.[illegible].svc	42.71	io (99.0%)	Web Request	esb-prd [illegible]	844.27
	41.44	io (91.0%)	Web Request	esb-prd [illegible]	845.40
	1.35	cpu (46.0%) io (54.0%)	RMI	esb-prd [illegible]	894.79
http://datasvcs.24hourfit.com[illegible]/CrmEntityData-1.0.0	3.07	io (86.0%)	Web Request	esb-prd [illegible]	899.50
	1.73	io (94.0%)	Web Request	esb-prd [illegible]	900.87
	1.39	cpu io (57.0%)	RMI	esb-prd [illegible]	906.80
http://datasvcs.24hourfit.com[illegible]/CrmEntityData-1.0.0	2.62	io (91.0%)	Web Request	esb-prd [illegible]	909.94
	1.42	io (95.0%)	Web Request	esb-prd [illegible]	911.13
	1.19	cpu io (72.0%)	RMI	esb-prd [illegible]	917.29
http://datasvcs.24hourfit.com[illegible]/CrmEntityData-1.0.0	4.61	io (96.0%)	Web Request	esb-prd [illegible]	919.47
	3.41	io (98.0%)	Web Request	esb-prd [illegible]	920.60
	1.10	cpu io (71.0%)	RMI	esb-prd [illegible]	926.43
http://datasvcs.24hourfit.com[illegible]/CrmEntityData-1.0.0	3.12	io (91.0%)	Web Request	esb-prd [illegible]	928.45
	1.86	io (96.0%)	Web Request	esb-prd [illegible]	929.68
https://24hourfitnessinc.com.[illegible].svc	747.25	io (100.0%)	Web Request	esb-prd [illegible]	942.70
	746.00	io (100.0%)	Web Request	esb-prd [illegible]	943.91

PROPRIETARY & CONFIDENTIAL

The road wasn't all smooth!

Challenges…

1. Marshalling and class loading

- **What** – Historically, the free version, Apache Ignite, didn't have the GridGain Binary Marshaller.
- **Issue** – Without the GridGain Binary Marshaller, our domain model classes (member, agreement etc..) would need to be loaded into each node in the cluster, and this didn't work when adding a node the cluster due to an issue in the version we were using.
- **Solution** - We created our own 24 Hour Custom Marshaller to serialize to standard java objects such that Ignite did not need our domain model classes.
- **Future** – We want to move away from our 24 Hour Custom Marshaller and instead use the Binary Marshaller that is now part of Apache Ignite.

2. Intermittent Node Connectivity

- **What** – We were setup with multicast and eagerly loading our employee data.
- **Issue** – We experienced intermittent issues with nodes not returning employee data.
- **Why** - Due to our network issues, nodes could not contact each other and thus nodes redistributed data. There was some data loss depending which nodes disconnect from each other and which node a particular call went to.
- **Solution** - We switched to TCP/IP where each server knows about all the IP addresses of other servers in the cluster.

3. Involve other groups/teams even though it is a software solution

- **Networking /Infrastructure** - Alluded to above, but additionally, we had timeouts that had the same affect of nodes exiting and rejoining the cluster.
- **Operations -** Instrumentation (for us Dynatrace), cache validation(Visor UI or scripts and REST APIs), reloading the cache (scripts and REST APIs).

Future possibilities

Previously shown future possibilities

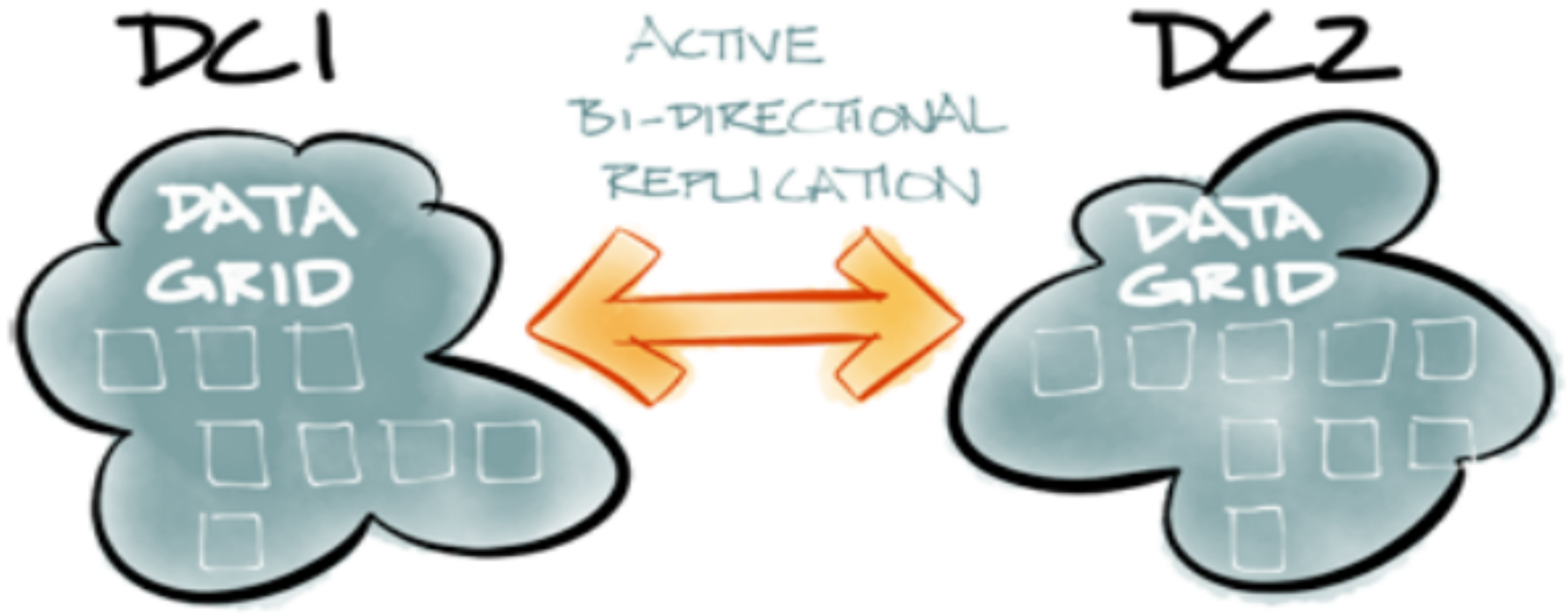
- Write Through
  - HR Employee data with database to support legacy systems
- Native Persistence
  - HR Employee data w/o database

Other ideas

- ID Generation vs. MySQL sequence
  - We have a vendor that has APIs where we are expected to provide their primary key IDs
  - It is a nightly batch with many rows/records so it would be nice/faster to use Ignite's in-memory ID generation
- Utilize the distributed nature of the Data Fabric to support CSA (Continuous Service Availability)
  - We currently have two data centers, and we are implementing an Active/Active solution across both data centers
    - When one data center is down for maintenance we'll utilize the other data center

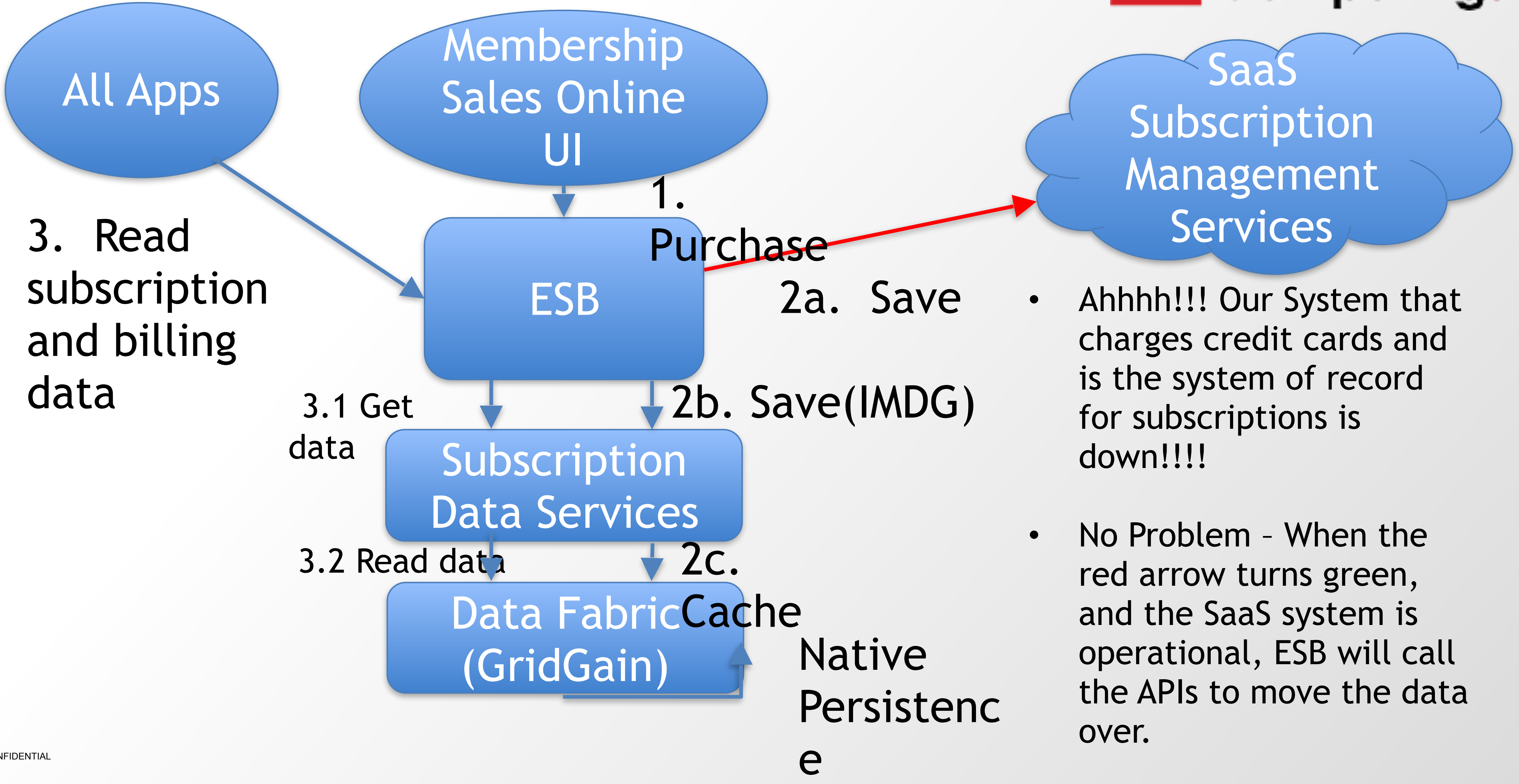
Future Possibility – Active/Active across our data centers

The picture below illustrates a simple view of Data Center Replication.



Future Possibility - Use Case 4: Transact while external systems are not available

Vendor has concurrency API rate limits, so use GridGain to lessen the load


All Apps
Membership Sales Online UI
1. Purchase
SaaS Subscription Management Services
3. Read subscription and billing data
ESB
2a. Save
3.1 Get data
2b. Save(IMDG)
Subscription Data Services
3.2 Read data
2c. Cache
Data Fabric (GridGain)
Native Persistence

- Ahhhh!!! Our System that charges credit cards and is the system of record for subscriptions is down!!!!
- No Problem – When the red arrow turns green, and the SaaS system is operational, ESB will call the APIs to move the data over.

Reasons why SaaS API Integrations aren’t Nirvana

1. **Not 24/7 (possibly specified in the contract allowing them to be down x hours per day/week/month, not that they actually are)**
2. **Not Performant (calls too slow, or chattiness makes business transaction too slow)**
3. **API Rate Limits (calls per minute or concurrency – shown below)**

This policy prevents tenants from monopolizing SaaS system resources.

Concurrent Request Limits

Each tenant has the following default concurrent request limits:

Type	Description	Default Limit for Concurrent, Uncompleted Requests	Retry After
Total Request	Refers to UI, REST, and SOAP API requests processed under a tenant. · API calls (REST and SOAP) ○ Create ○ Update ○ Get ○ Delete ○ Generate ○ Query · UI requests (all) Excluded from this policy · Login API calls (REST and SOAP) · UI login requests	40	120 seconds

Summary

What have we seen?

- 24 Hour Fitness' historical architecture and some limitations
- How in-memory solutions solved use cases at 24 Hour Fitness
- Issues we ran into, marshalling and intermittent node connectivity
- How in-memory solutions might be leveraged in the future at 24 Hour Fitness

Tidbits

- A thank you to the Target IMC Summit 2016 presenters
  - Maybe you'll be up here next year, and I'll be watching you present!
- Careers at 24 Hour Fitness IT
  - North County San Diego (Carlsbad)
  - http://careers.24hourfitness.com
- A thank you to all my colleagues at 24 Hour Fitness that helped me with details in this presentation, attended my practice runs, and to those who implemented these in-memory solutions

Questions…..

Craig Gresbrink

cgresbrink@24hourfit.com

– Subject – IMC (start with IMC)
  - I will do my best to respond within 2 or 3 days, for at least the next month.

Thank You!

Don't forget to check out the next slide: References and further reading…

References and further reading

Give credit, where credit is due…

- Hazelcast
  - https://hazelcast.com/
- Cache-Aside Pattern
  - https://blog.cdemi.io/design-patterns-cache-aside-pattern/
- You Are Not Netflix: How and When to Use Microservices in the Enterprise
  - https://www.gartner.com/webinar/3437517
- Microservices vs. Service Oriented Architecture
  - http://www.oreilly.com/programming/free/microservices-vs-service-oriented-architecture.csp
- Apache Ignite
  - https://ignite.apache.org/
- Apache Ignite's Read Through, Write Through, and Write Behind
  - https://apacheignite.readme.io/v1.1/docs/persistent-store
- Apache Ignite Native Persistence
  - https://apacheignite.readme.io/docs/distributed-persistent-store
- Camunda - an open source platform for workflow and business process management
  - https://camunda.org/
- Dynatrace - an Automated Performance Management product (APM)
  - https://www.dynatrace.com/
- Apache Ignite's Binary Marshaller
  - https://apacheignite.readme.io/docs/binary-marshaller
- Apache Ignite's ID Generator
  - https://apacheignite.readme.io/docs/id-generator
- GridGain's Data Center Replication
  - https://docs.gridgain.com/v8.1/docs/data-center-replication
- IMC Summit 2016 - Target's Presentation which was my inspiration (Thank You)
  - https://www.imcsummit.org/2016/videos-and-slides/targets-first-foray-into-an-in-memory-data-grid-and-the-trips-stumbles-and-falls-that-came-with/
- High performance in-memory computing with Apache Ignite (I haven't read this but it looked good)
  - https://www.amazon.com/Performance-memory-computing-Apache-Ignite/dp/1365732355/ref=sr_1_cc_1?s=aps&ie=UTF8&qid=1507758595&sr=1-1-catcorr

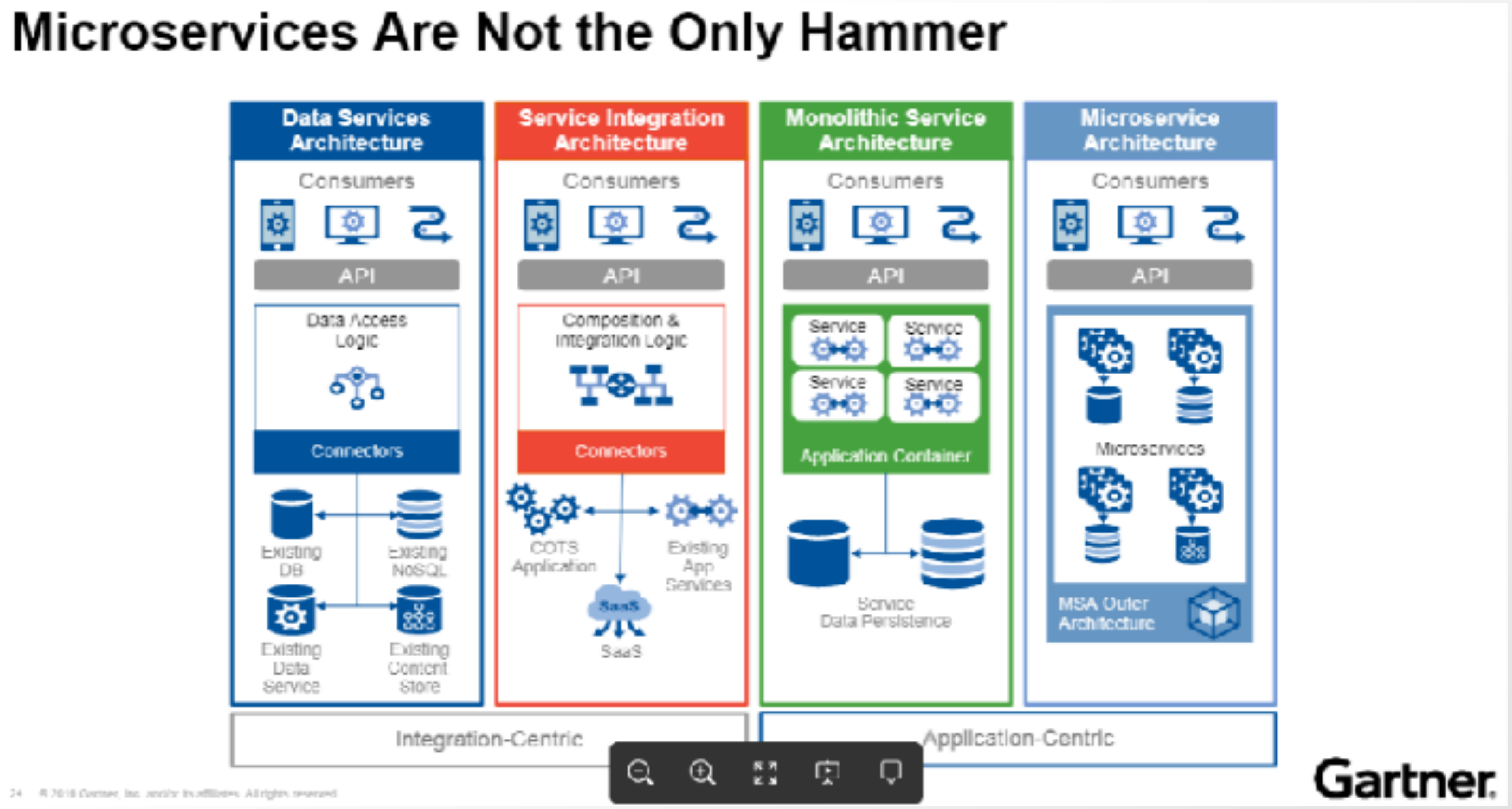
Bonus slides

»B-SIDES

»OUTTAKES

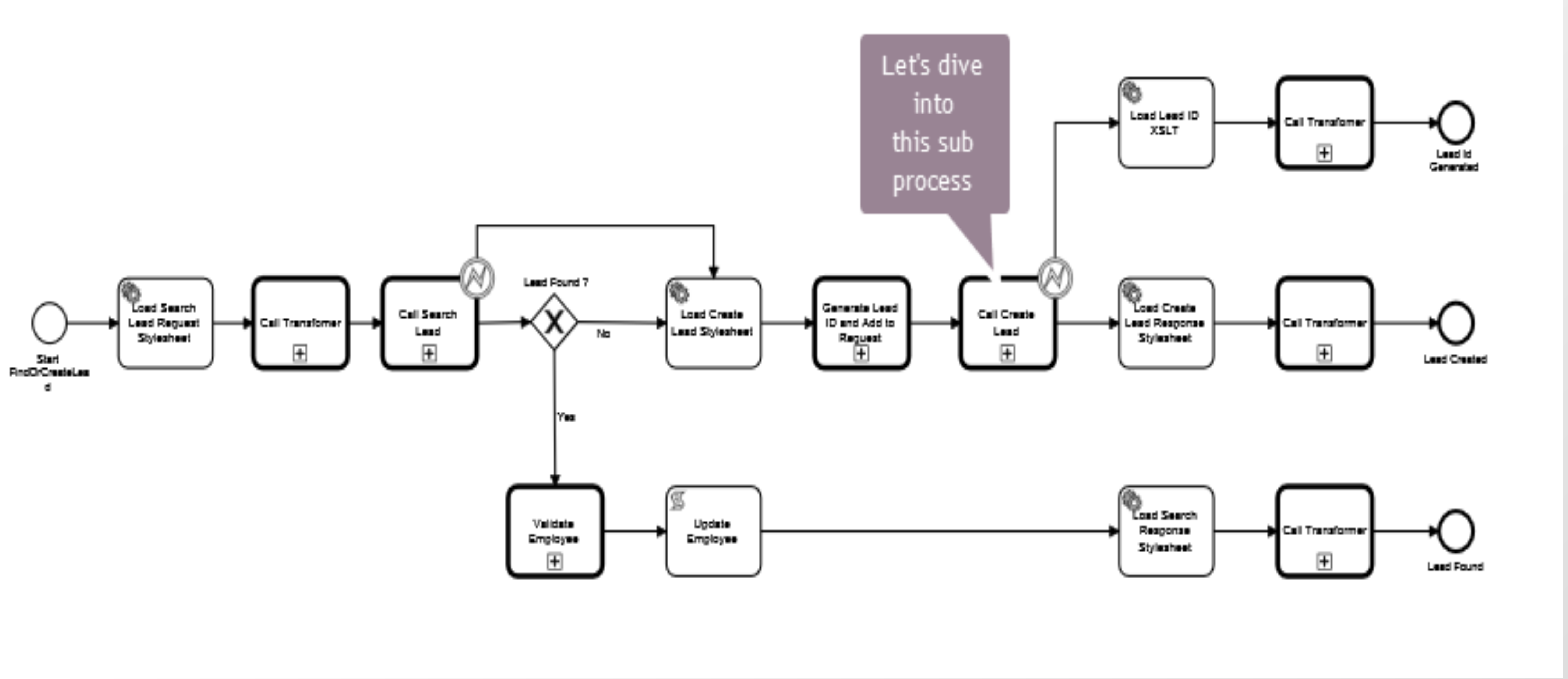
The following slides were contemplated but didn't make the final cut....

What is this, 2005, ESB? No Microservices?



Use case 3: A chatty real-time integration with CRUD-based APIs is too slow

What does lead creation look like in a visual Business Process Management System?



Camunda