In-Memory Computing SUMMIT
NORTH AMERICA 2019

GridGain Ultimate Edition aids implementation of SaaS systems and replaces traditional databases

Craig Gresbrink

Solutions Architect
24 Hour Fitness

Proud Sponsor of the U.S. Olympic & Paralympic Teams

YOUR RESULTS. YOUR WAY.™

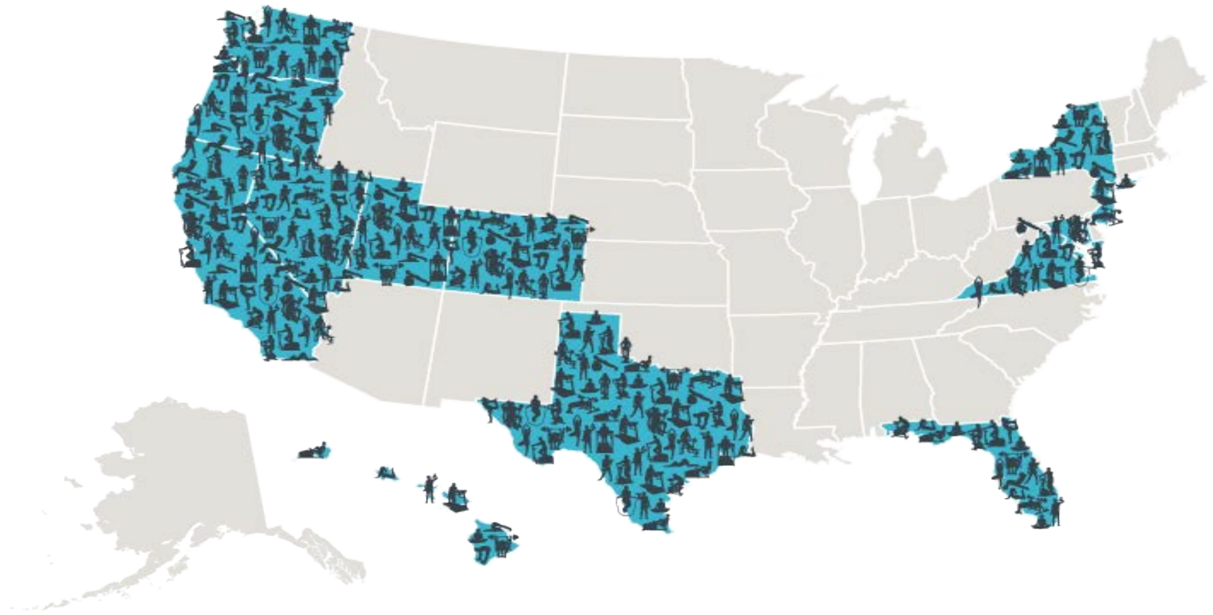
Who, What, Why, How, and Learnings

Tales from the trenches

In-Memory Computing SUMMIT | NORTH AMERICA 2019

Who are we?

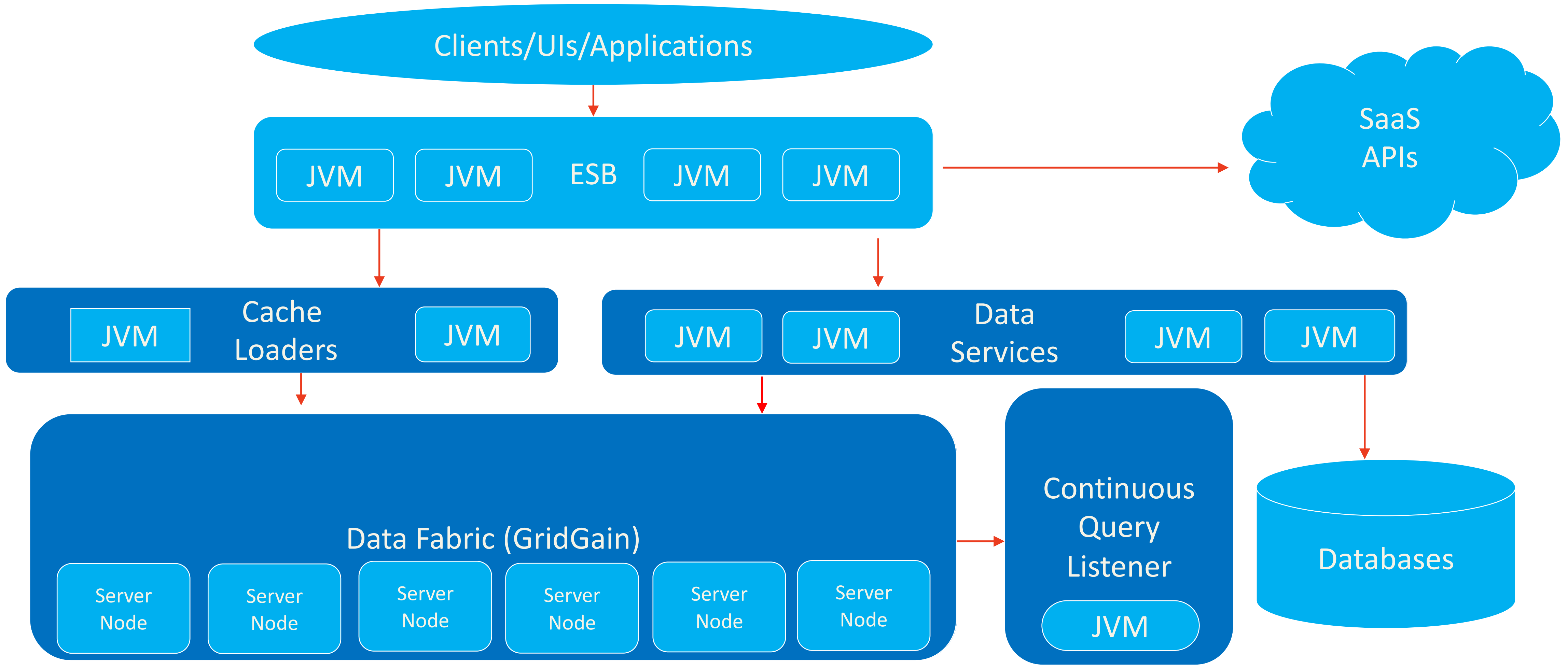
24 Hour Fitness is a leading fitness industry pioneer with more than **400 clubs** across the United States. 24 Hour Fitness has **20,000 plus employees** serving nearly **4 million** club members.



What we built

Architecture and infrastructure

Current Architecture and “The Grid”



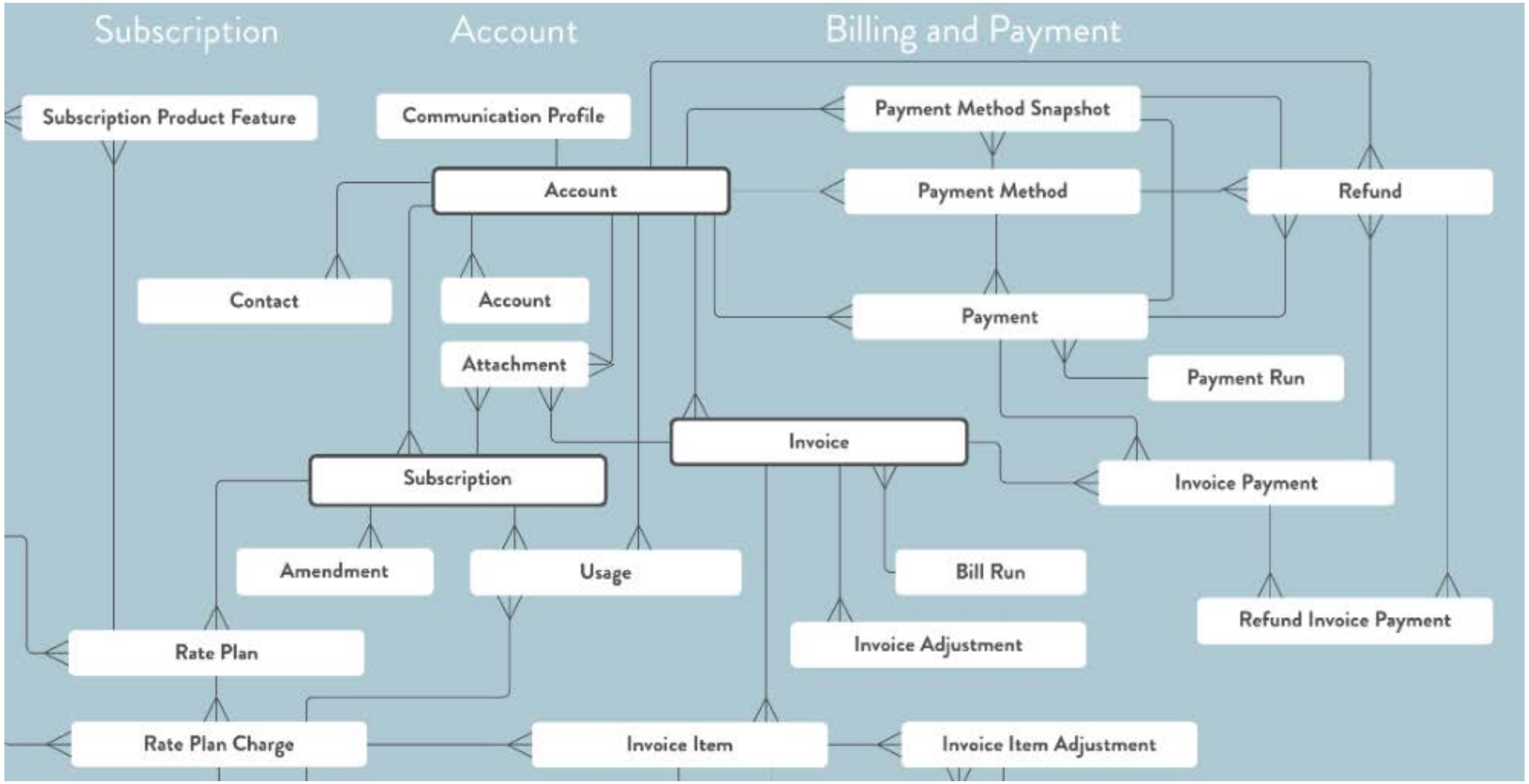
Infrastructure:
- not my specialty, but very important

7 Node GridGain Cluster…

Each Virtual Server Node:

- 8 vCPUs
- 64GB RAM
  - Memory segmentation
    - *45G Durable*
    - *14G Java Heap*
    - *5G OS/Disk Buffers/GG Disk Pointers*

Billing System Object Model



YOUR RESULTS. YOUR WAY.™

In-Memory Computing SUMMIT NORTH AMERICA 2019

Volume of Data

1.5 million (member) accounts

1.8 million Subscriptions

2 million Rate plans

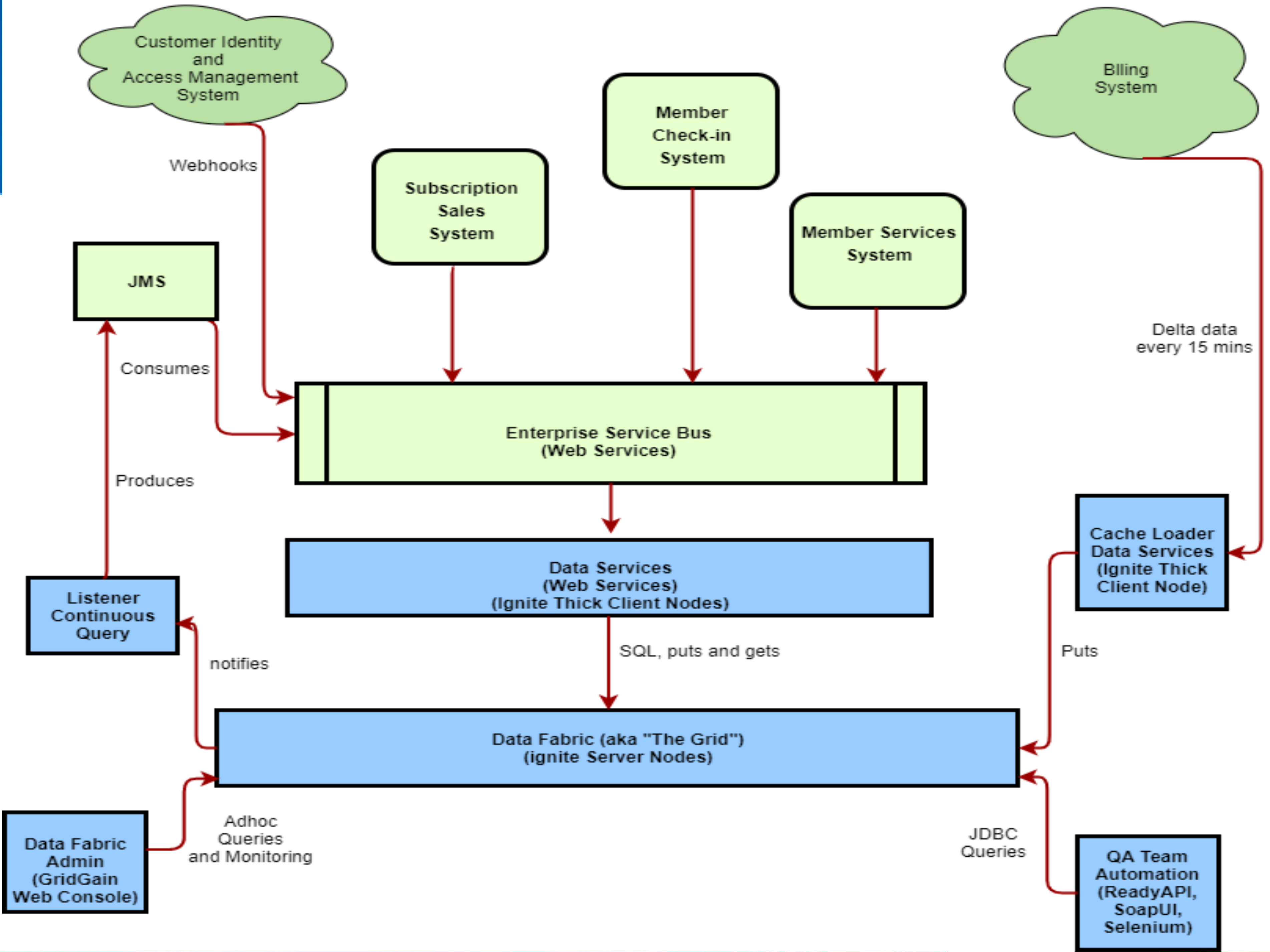
8.2 million Rate plan charges

3 million Invoices

9 million Invoice items (and will continue to grow rapidly)

1.4 million payments

High level



Customer Identity and Access Management System
Webhooks
Subscription Sales System
Member Check-in System
Member Services System
Billing System
JMS
Consumes
Produces
Enterprise Service Bus (Web Services)
Delta data every 15 mins
Data Services (Web Services) (Ignite Thick Client Nodes)
Cache Loader Data Services (Ignite Thick Client Node)
Listener Continuous Query
notifies
SQL, puts and gets
Puts
Data Fabric (aka "The Grid") (ignite Server Nodes)
Adhoc Queries and Monitoring
JDBC Queries
Data Fabric Admin (GridGain Web Console)
QA Team Automation (ReadyAPI, SoapUI, Selenium)

YOUR RESULTS. YOUR WAY.™

Why we built it

External Systems' API constraints

Why use an In-Memory Data Grid?

Some Cloud/SaaS APIs:

1. Are slow and chatty
   - a double edge sword as this means, extra slow
2. Are not guaranteed to be up 24/7
3. Have rate limits
4. Can't support searches
   - LastName like 'Gre%'
5. Allow for single object/entity/table querying only
   - Lack of joins, hampers development, debugging and production support

Single Data Store for data from disparate systems

Wait, a traditional database can solve this!, but the grid is…

IMDGs are fast and scalable

- not just fast, but really fast, as we shall see.

YOUR RESULTS. YOUR WAY.™

How did we build it

Cache Loaders

Data Services and SQL Queries

Continuous Queries

In-Memory Computing SUMMIT NORTH AMERICA 2019

GridGain Support – 40 Days

Do you want to refactor?

Depending on timelines and your appetite for trial and error, consider using GridGain support

Q3 2017

2017 IMC Presentation involves a vision of caching billing system data as a solution to API rate limits

Q1/Q2 2018

Evangelizing the idea, which is at first dismissed as unnecessary now, maybe later

Q3 2018

Like clauses in member search expose need for a solution. 40 day support contract signed to build "The Grid"

Q2 2019

2 Club "pilot" goes live

Q3/Q4 2019

20, 40, 100 clubs clubs/600k accounts:

exposes **full table scans = SQL Tuning**

ALL **430 Clubs!**

SQL Platform - Cache Loaders hold Cache/SQL Config

```
<bean class="org.apache.ignite.configuration.CacheConfiguration">
    <property name="name" value="SUBSCRIPTION_CACHE" />
    …
    <property name="queryEntities">
        <property name="fields">
             <map>
                 <entry key="accountId" value="java.lang.String" />
                          …
        <property name="indexes">
            <list>
                <bean class="org.apache.ignite.cache.QueryIndex">  <constructor-arg value="id" />   </bean>
                <bean class="org.apache.ignite.cache.QueryIndex">  <constructor-arg value="accountId" />  </bean>
```

YOUR RESULTS. YOUR WAY.™

In-Memory Computing SUMMIT NORTH AMERICA 2019

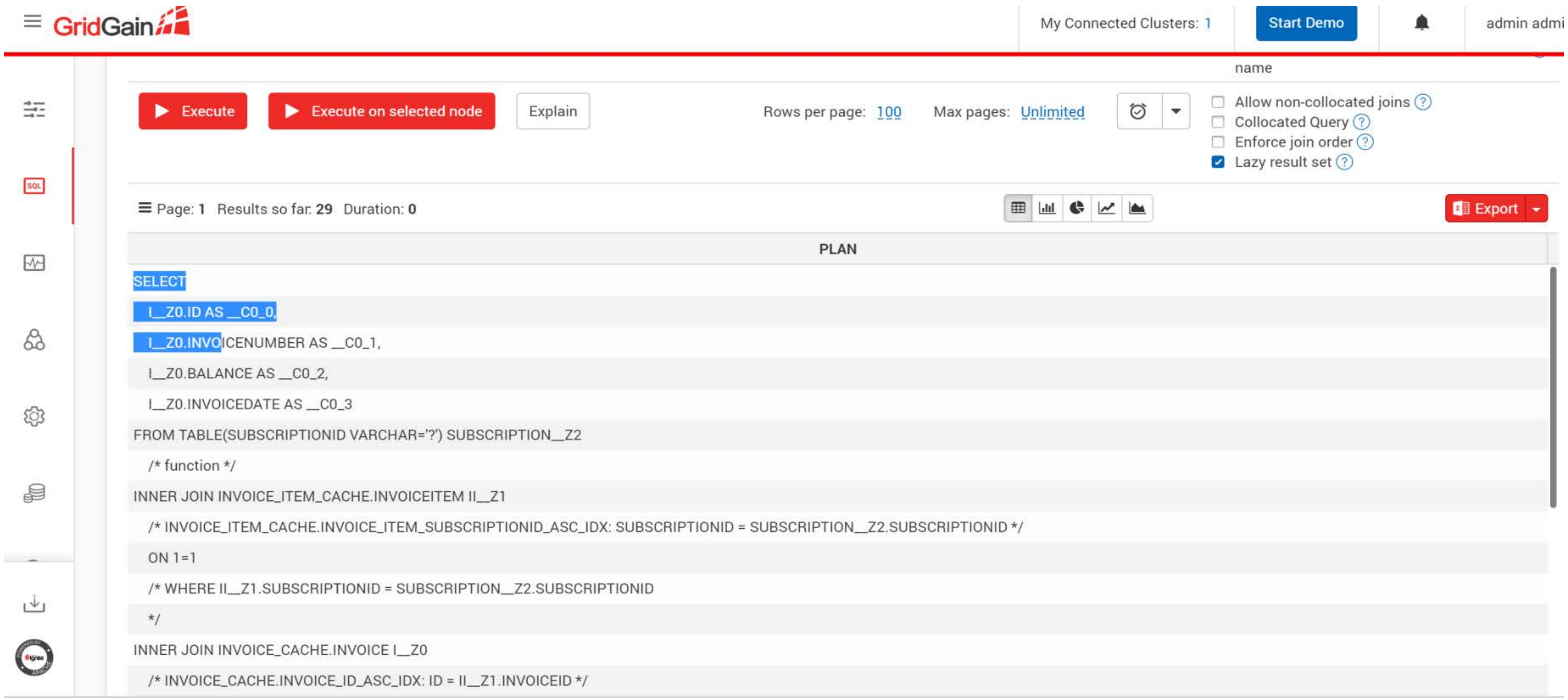
SQL Platform – Data Services Ignite Thick Clients Run Queries

```
List<Row> openInvoices =
dataFabricFacadeEjb.runAttributesQuery(QueryFactory.getOpenInvoicesBySubscriptionIds (), new Object[] {
subscriptionIds.toArray() });



public static Query getOpenInvoicesBySubsQuery() {

  return new Query(

“SELECT i.id,
           i.invoicenumber,
           i.balance,
           i.invoicedate
FROM invoice_cache.invoice i
            INNER JOIN invoice_item_cache.invoiceitem ii ON i.id = ii.invoiceId
            JOIN table(subscriptionid varchar = ? ) subscription ON ii.subscriptionid = subscription.subscriptionId
WHERE i.posteddate IS NOT NULL
     AND i.balance > 0
GROUP BY i.id,
                     i.invoicenumber,
                     i.balance;", false);  }
```

YOUR RESULTS. YOUR WAY.

In-Memory Computing SUMMIT NORTH AMERICA 2019

SQL Platform – Web Console, Explain Plans



Rows per page: 100   Max pages: Unlimited

- [ ] Allow non-collocated joins
- [ ] Collocated Query
- [ ] Enforce join order
- [x] Lazy result set

Page: 1 Results so far: 29 Duration: 0

PLAN

```
SELECT
  I__Z0.ID AS __C0_0,
  I__Z0.INVOICENUMBER AS __C0_1,
  I__Z0.BALANCE AS __C0_2,
  I__Z0.INVOICEDATE AS __C0_3
FROM TABLE(SUBSCRIPTIONID VARCHAR='?') SUBSCRIPTION__Z2
  /* function */
INNER JOIN INVOICE_ITEM_CACHE.INVOICEITEM II__Z1
  /* INVOICE_ITEM_CACHE.INVOICE_ITEM_SUBSCRIPTIONID_ASC_IDX: SUBSCRIPTIONID = SUBSCRIPTION__Z2.SUBSCRIPTIONID */
  ON 1=1
  /* WHERE II__Z1.SUBSCRIPTIONID = SUBSCRIPTION__Z2.SUBSCRIPTIONID
  */
INNER JOIN INVOICE_CACHE.INVOICE I__Z0
  /* INVOICE_CACHE.INVOICE_ID_ASC_IDX: ID = II__Z1.INVOICEID */
```

```
  AND (I__Z0.ID = II__Z1.INVOICEID))
GROUP BY I__Z0.ID, I__Z0.INVOICENUMBER, I__Z0.BALANCE
```

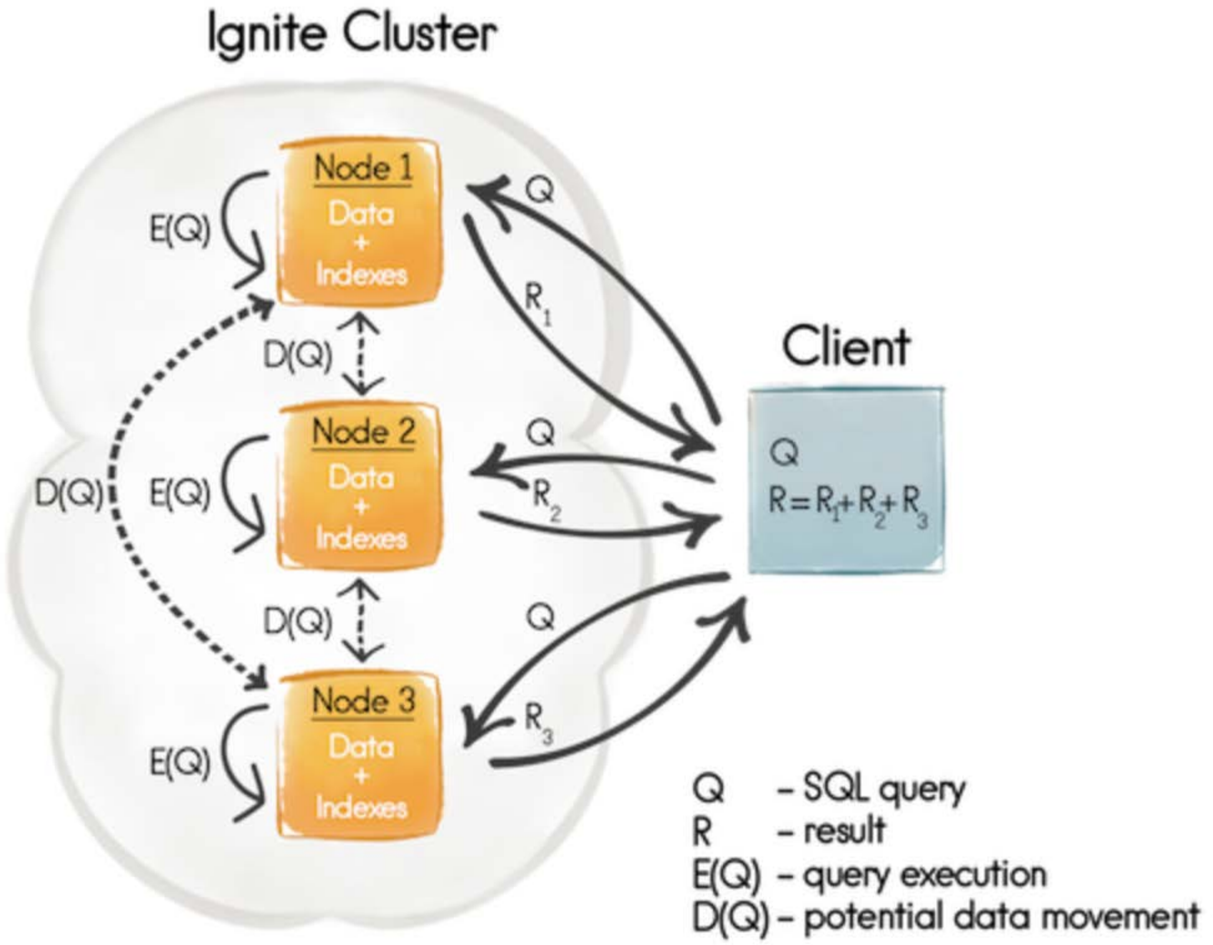
YOUR RESULTS. YOUR WAY.™

In-Memory Computing SUMMIT NORTH AMERICA 2019

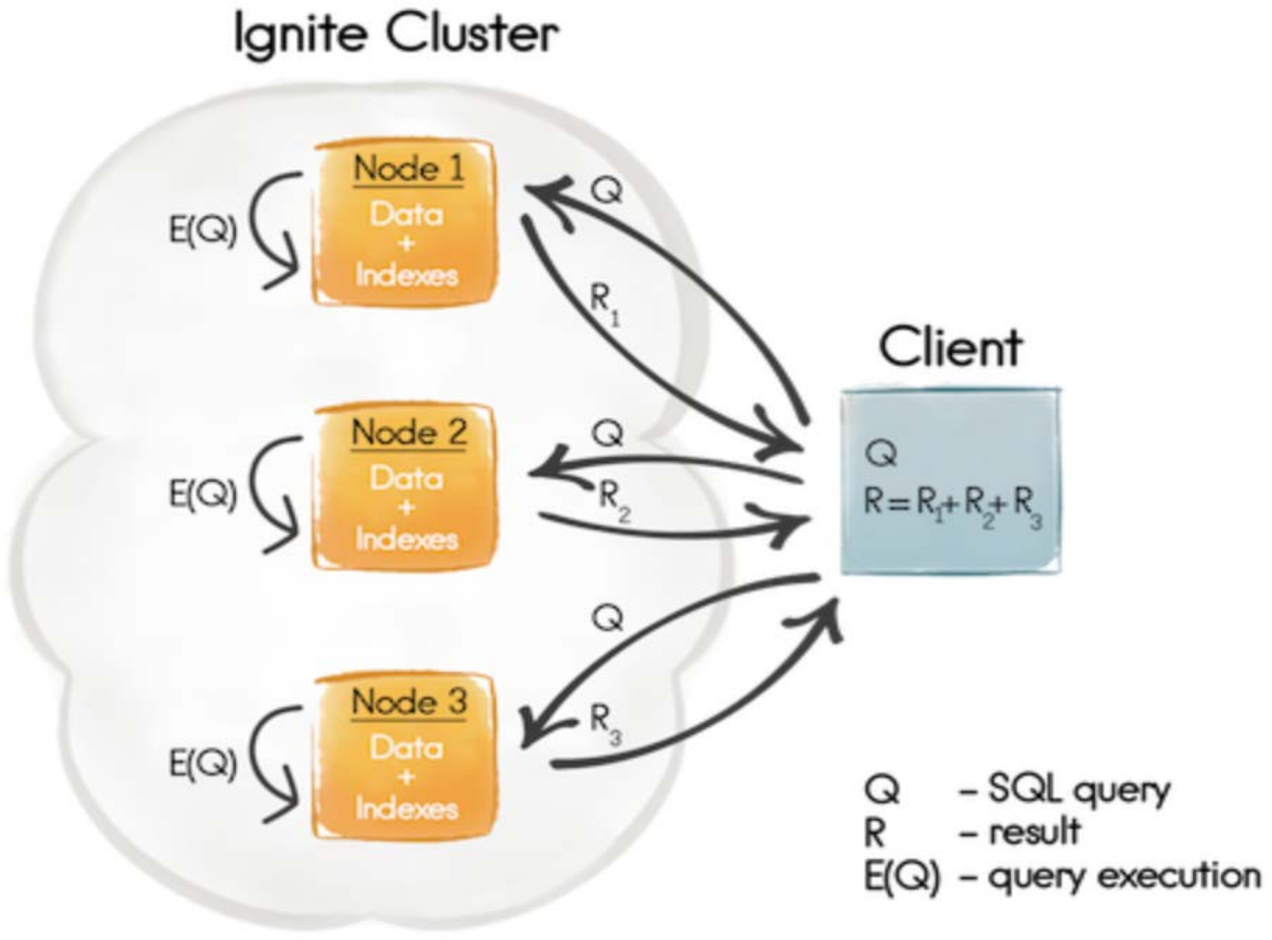
Affinity Domains

Avoid **distributedJoins**=true which is synonymous with checking **Allow non-collocated joins**

To allow DistributedJoins=true or not?

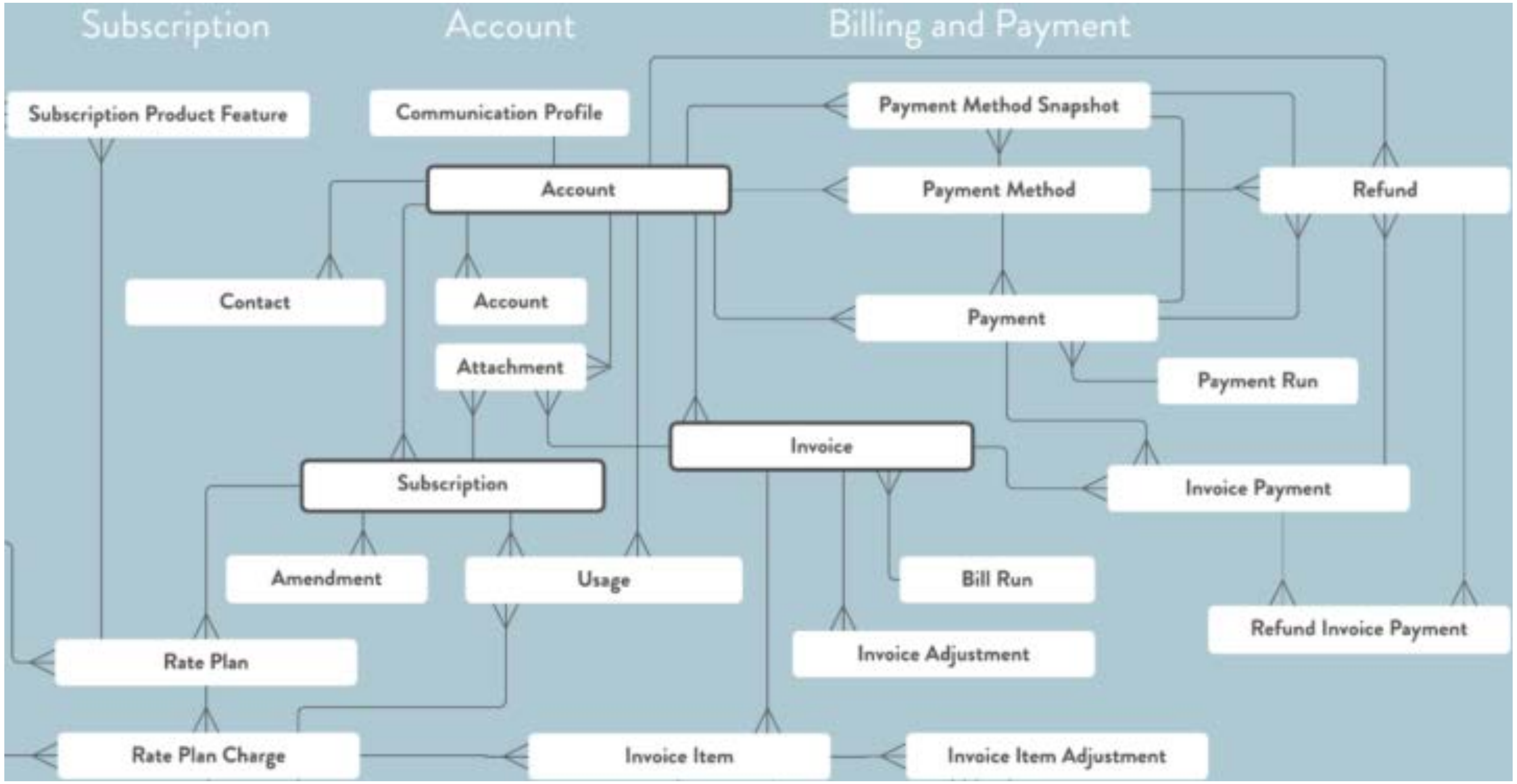


Picture 2. Non-collocated SQL Query

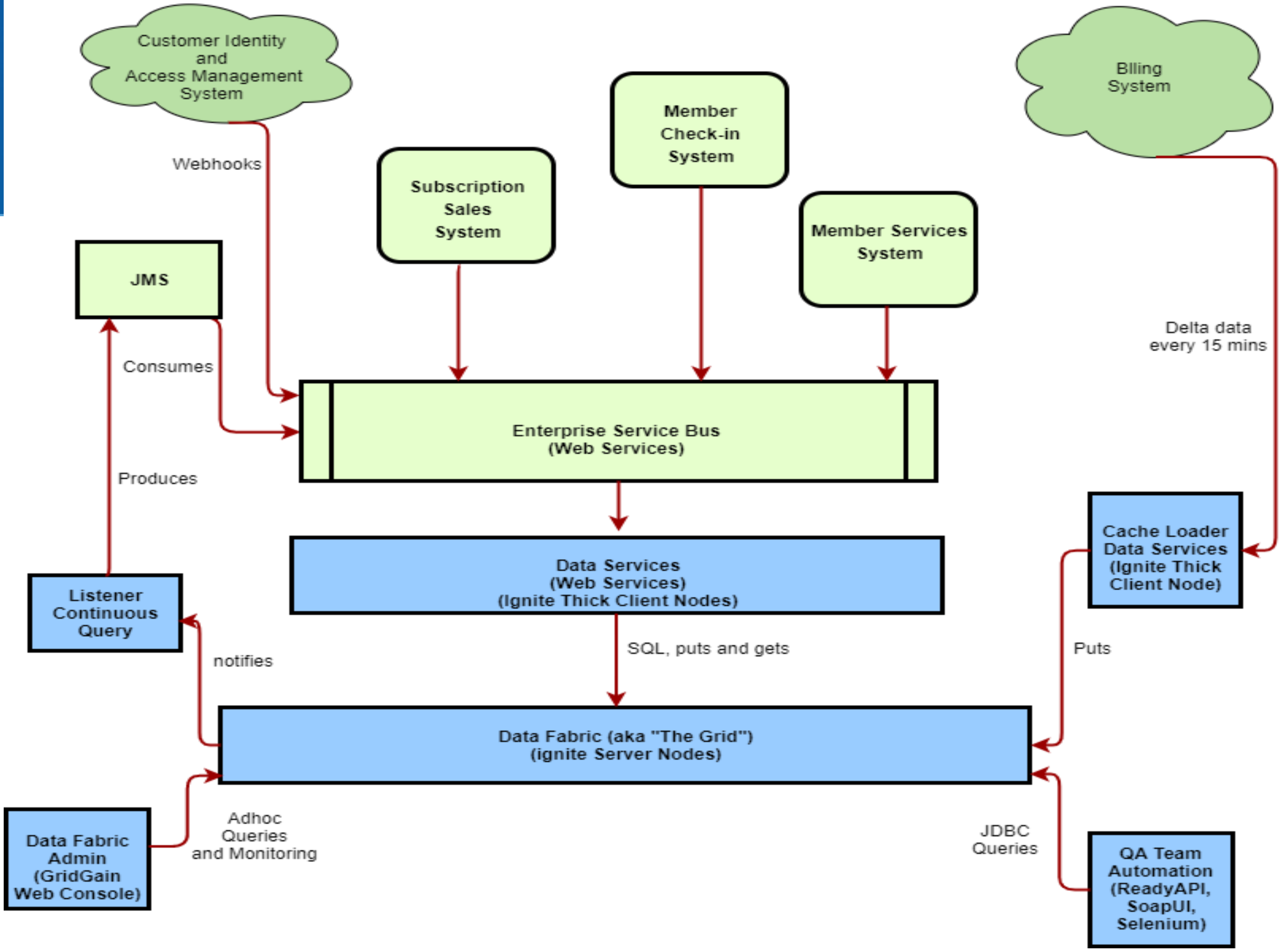


Picture 1. Collocated SQL Query

Billing System Object Model

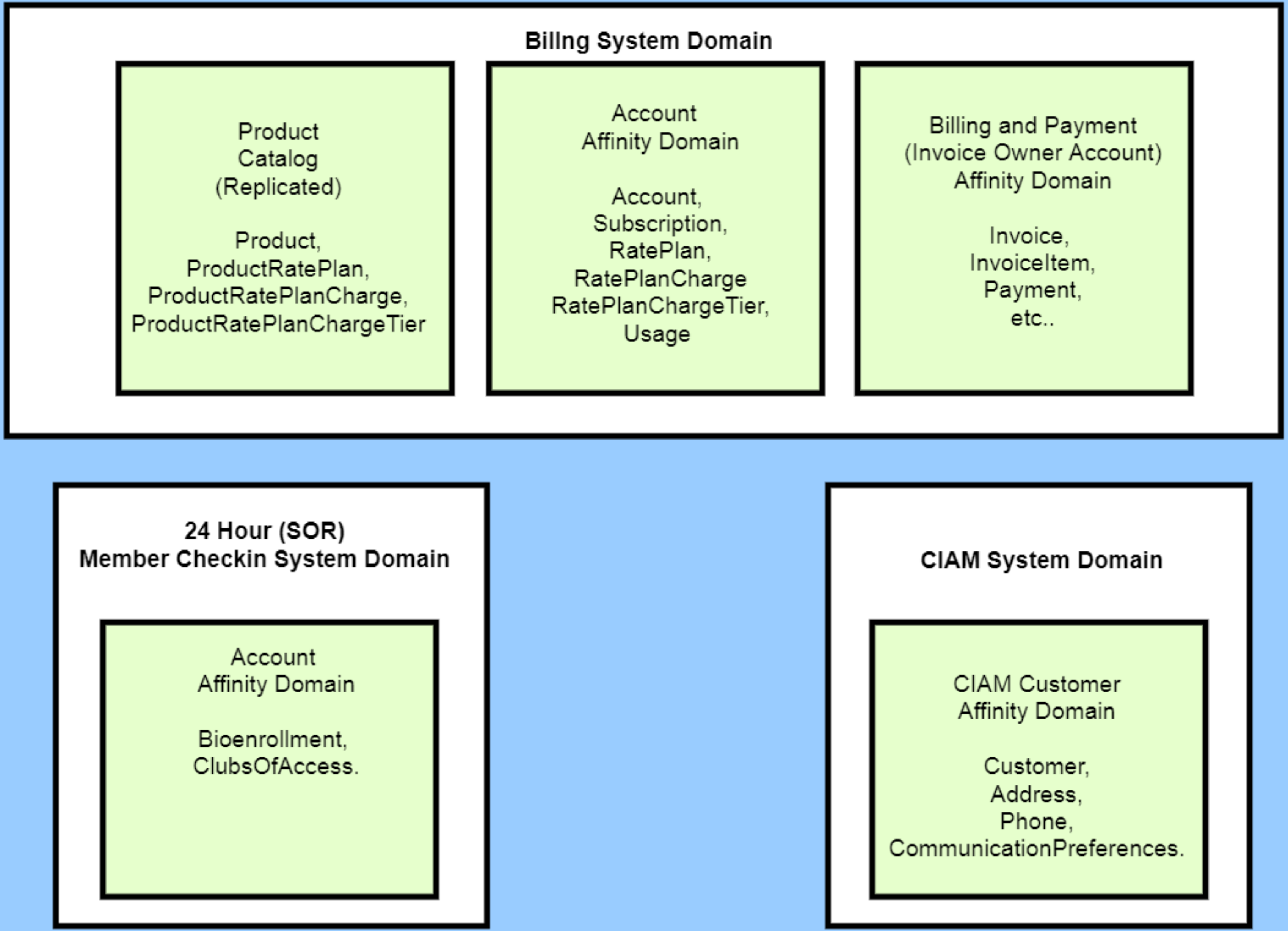


What data is in the grid?



Customer Identity and Access Management System
Webhooks
Subscription Sales System
Member Check-in System
Member Services System
Billing System
JMS
Consumes
Produces
Enterprise Service Bus (Web Services)
Delta data every 15 mins
Cache Loader Data Services (Ignite Thick Client Node)
Data Services (Web Services) (Ignite Thick Client Nodes)
Listener Continuous Query
notifies
SQL, puts and gets
Puts
Data Fabric (aka "The Grid") (ignite Server Nodes)
Data Fabric Admin (GridGain Web Console)
Adhoc Queries and Monitoring
JDBC Queries
QA Team Automation (ReadyAPI, SoapUI, Selenium)

24 Hour Fitness Affinity Domains



YOUR RESULTS. YOUR WAY.™

In-Memory Computing SUMMIT NORTH AMERICA 2019

It’s a SOR subject!

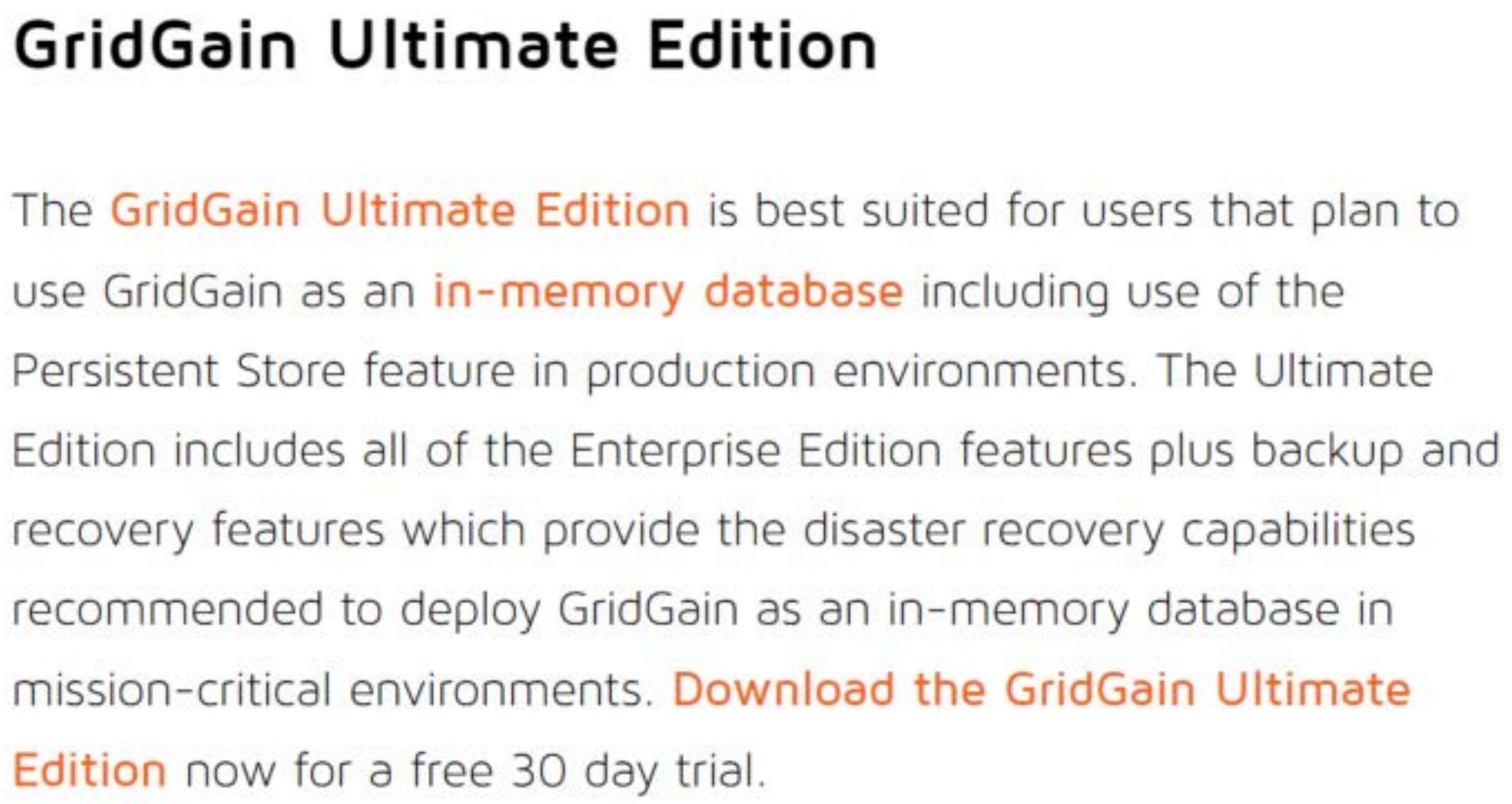
System **O**f **R**ecord

1. Inserts and updates occur to the grid and only to the grid

This leads to, as it would for any database:

1. How do I back it up?
2. How do I restore it?

Ultimately, you need Ultimate



GridGain Ultimate Edition

The GridGain Ultimate Edition is best suited for users that plan to use GridGain as an in-memory database including use of the Persistent Store feature in production environments. The Ultimate Edition includes all of the Enterprise Edition features plus backup and recovery features which provide the disaster recovery capabilities recommended to deploy GridGain as an in-memory database in mission-critical environments. Download the GridGain Ultimate Edition now for a free 30 day trial.

Why use Ultimate?

1. The Grid is a System of Record acting as an IMDB

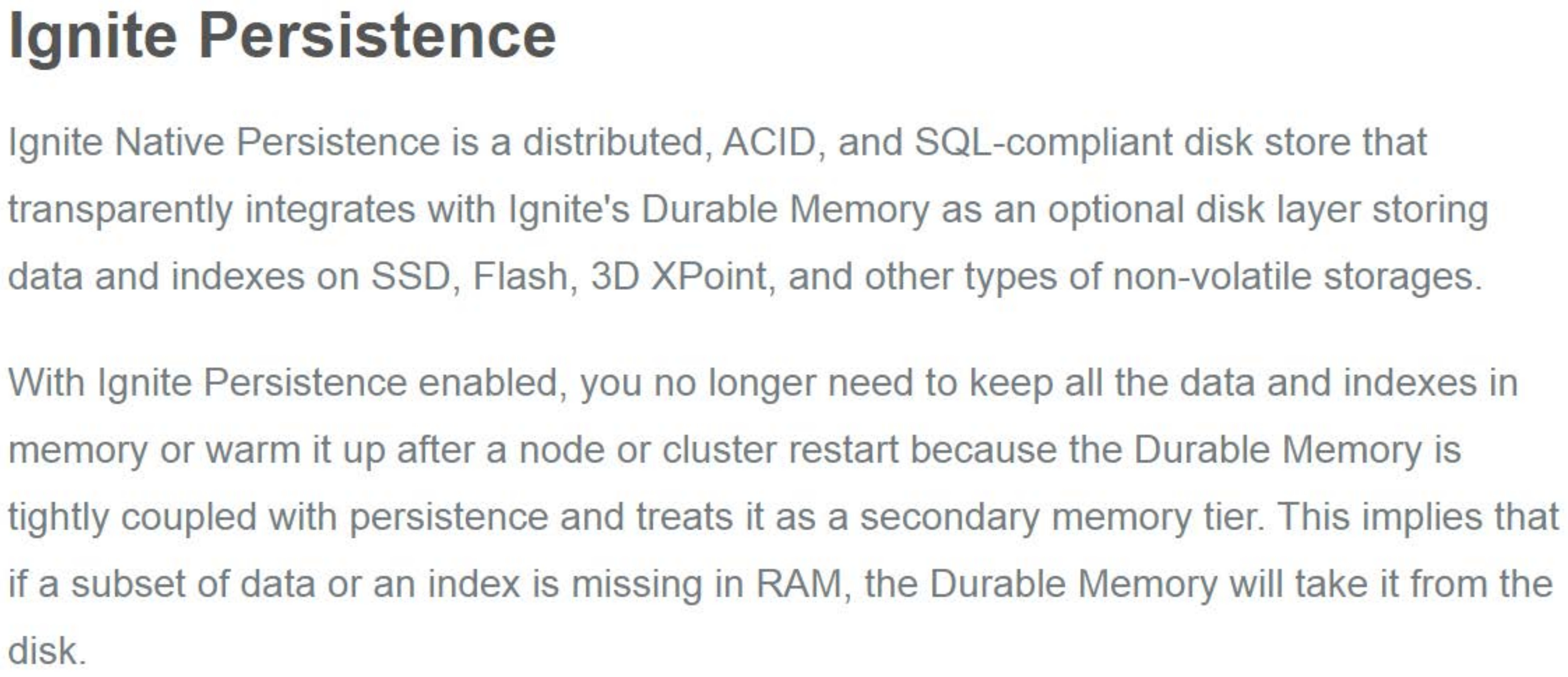
- This is the case for member check-in data

OR

2. Data volumes in your source system are “too large/slow” to retrieve and repopulate the grid.

  - This is the case for invoices and payments in our billing system
  - Restoration of The Grid, via Ultimate, is much faster, and much more reliable, since you don’t depend on external systems.

Native persistence



Ignite Persistence

Ignite Native Persistence is a distributed, ACID, and SQL-compliant disk store that transparently integrates with Ignite's Durable Memory as an optional disk layer storing data and indexes on SSD, Flash, 3D XPoint, and other types of non-volatile storages.

With Ignite Persistence enabled, you no longer need to keep all the data and indexes in memory or warm it up after a node or cluster restart because the Durable Memory is tightly coupled with persistence and treats it as a secondary memory tier. This implies that if a subset of data or an index is missing in RAM, the Durable Memory will take it from the disk.

Ultimate – Backup and Recovery

24 Hour Fitness:

1. Nightly full backups/snapshots
2. Hourly incremental backups (deltas)

You can do point in time recovery up to the minute via the WAL – Write Ahead Log

We retain these backups for 5 days

You'll need to consider what makes sense for your business

Continuous Query Payment of an invoice

User Story:

As a member services representative

I want to have the system send a thank you email/text to all members when they make an on time payment of their invoice

So that we show appreciation for their business and the member receives confirmation of the transaction

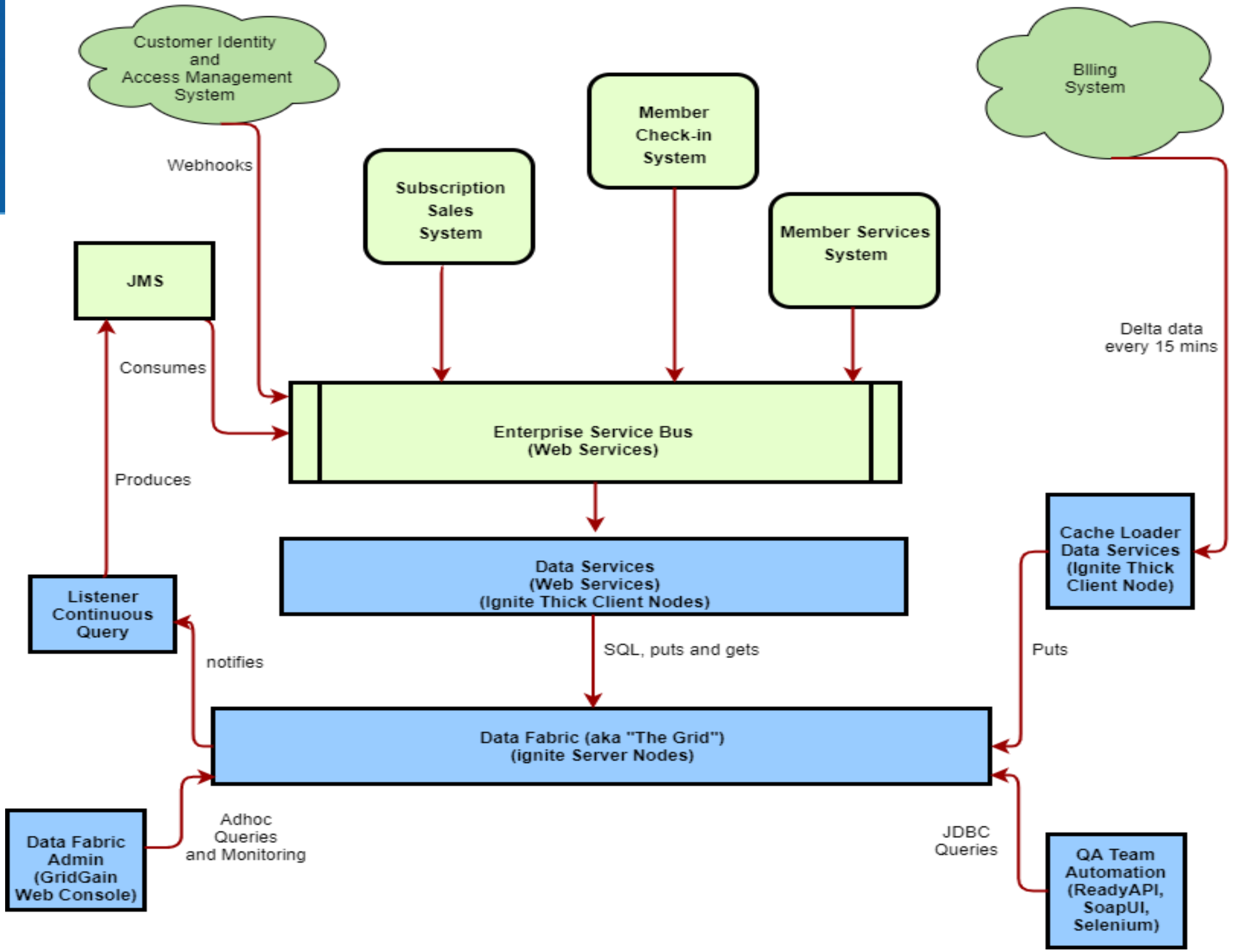
Given that the In-Memory Data Grid receives payment information every 15 minutes

And that it also has their email, and possibly their mobile phone and opt-in/out to text messages

When we determine their communication preferences

Then we send them an email (if opted in) and/or a text (if opted in)

Remote Filters and Local Listeners


Customer Identity and Access Management System
Webhooks
Subscription Sales System
Member Check-in System
Member Services System
Blling System
JMS
Consumes
Produces
Delta data every 15 mins
Enterprise Service Bus (Web Services)
Data Services (Web Services) (Ignite Thick Client Nodes)
Cache Loader Data Services (Ignite Thick Client Node)
Listener Continuous Query
notifies
SQL, puts and gets
Puts
Data Fabric (aka "The Grid") (ignite Server Nodes)
Data Fabric Admin (GridGain Web Console)
Adhoc Queries and Monitoring
JDBC Queries
QA Team Automation (ReadyAPI, SoapUI, Selenium)

YOUR RESULTS. YOUR WAY.

In-Memory Computing SUMMIT NORTH AMERICA 2019

Learnings

"Optimizer", Explain Plans, and SQL Tuning

Indexes, if 1 is good more is better!

What does the documentation say?:

Indexes Tradeoffs

There are multiple things you should consider when choosing indexes for your Ignite application.

Indexes are not free. They consume memory, also each index needs to be updated separately, thus your cache update performance can be poorer when you have more indexes set up. On top of that**, the optimizer might do more mistakes by choosing a wrong index to run a query**.

It is a bad strategy to index everything!

Pain is just weakness leaving the body!

What worked in Oracle will work in the Grid, right?

- Naïve approach, create indexes on status columns etc… to support potential adhoc queries

Learning:

The Grid's optimizer is not as sophisticated as Oracle's optimizer.

A new 24 Hour Fitness standard emerges

New Standards emerge:

1. Create indexes on these, and only these:
   - *Ids(primaryKeys) ex. Accound.id, Subscription.id*
   - *Foreign keys (for joins) ex. Subscription.accountId*
   - *Natural keys (unique or not) ex. Account.accountNumber, Subscription.name*
   - *Dates (for batch jobs that query on date ranges) – Account.updatedDate*
     - Be careful, it might start using your batch/date indexes if that column is in your transactional queries.

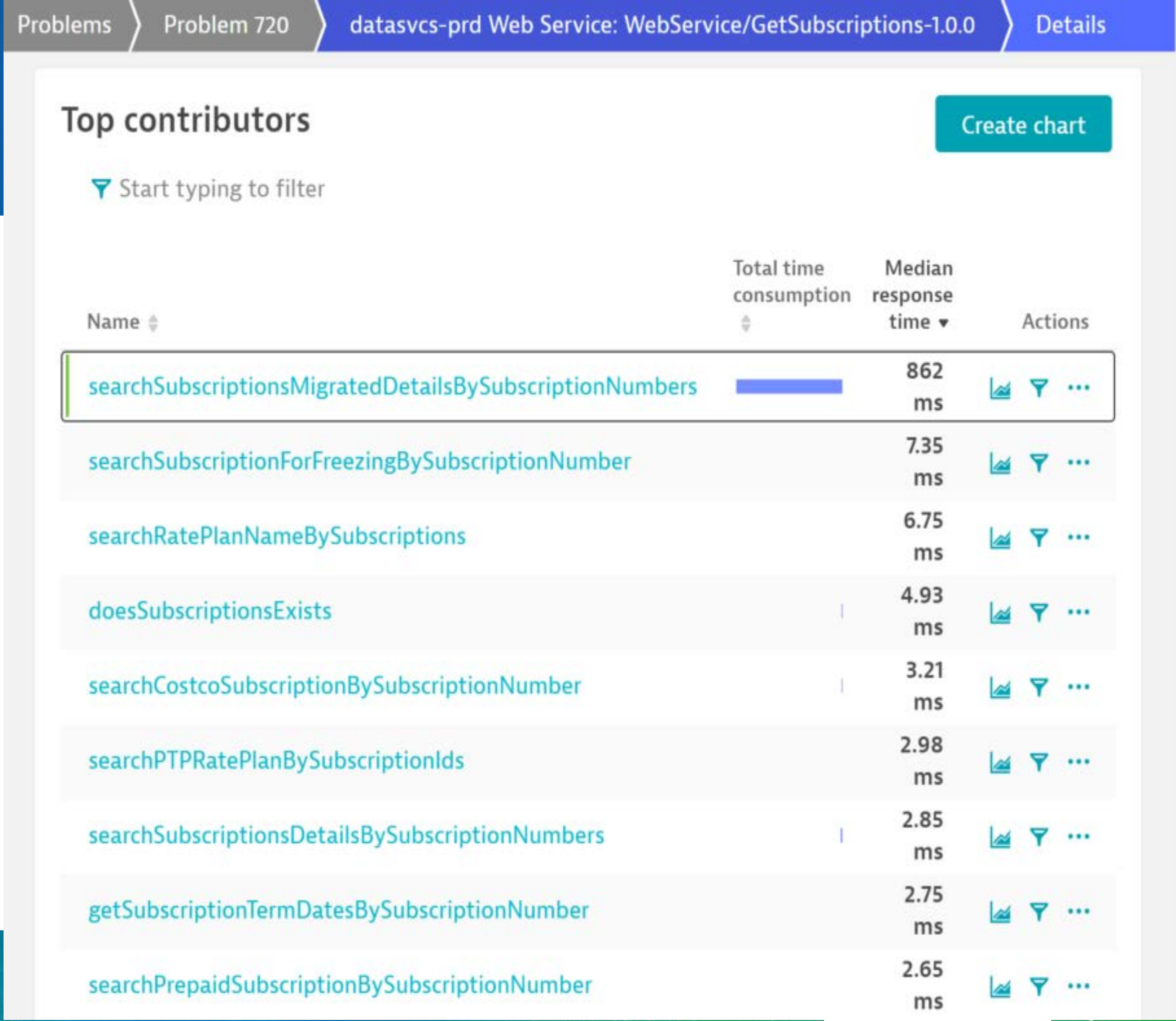
2. Explain plans must be presented for any new/changed queries
   - They are “code” reviewed by “The Cache Team” to verify the proper indexes are being used.

Can I get a hint?

Sometimes the optimizer chooses the wrong index, so help the optimizer help you:

```
SELECT invoice.duedate, invoice.invoicenumber, invoiceitem.quantity,
invoiceitem.uom, invoiceitem.unitprice, invoice.balance, invoice.id
FROM invoice_item_cache.invoiceitem USE
INDEX(INVOICE_ITEM_CACHE.INVOICE_ITEM_SUBSCRIPTIONID_ASC_IDX)
INNER JOIN invoice_cache.invoice ON invoice.id = invoiceitem.invoiceid
JOIN TABLE(id varchar = ( '123' )) s ON s.id = invoiceitem.subscriptionid
WHERE invoice.balance = 0
AND invoiceitem.uom = 'Session'
AND invoice.duedate < sysdate
AND invoice.invoicedate < sysdate
AND invoice.reversed <> true
AND ( invoiceitem.chargename = 'Personal Training Fee' OR
invoiceitem.chargename LIKE 'Personal Training Fee%' )
GROUP BY invoice.id ORDER BY invoice.duedate
```

APM Timings



Problems > Problem 720 > datasvcs-prd Web Service: WebService/GetSubscriptions-1.0.0 > Details

Top contributors

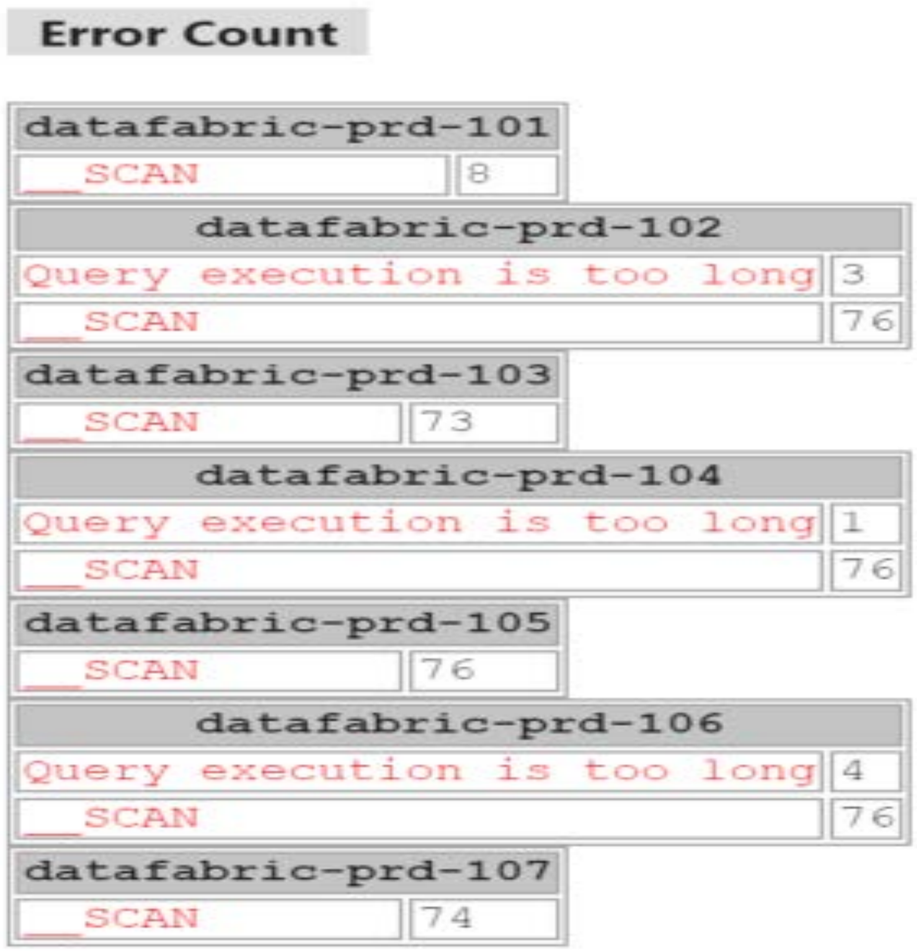
Create chart

Start typing to filter

Name	Total time consumption	Median response time	Actions
searchSubscriptionsMigratedDetailsBySubscriptionNumbers		862 ms	
searchSubscriptionForFreezingBySubscriptionNumber		7.35 ms	
searchRatePlanNameBySubscriptions		6.75 ms	
doesSubscriptionsExists		4.93 ms	
searchCostcoSubscriptionBySubscriptionNumber		3.21 ms	
searchPTPRatePlanBySubscriptionIds		2.98 ms	
searchSubscriptionsDetailsBySubscriptionNumbers		2.85 ms	
getSubscriptionTermDatesBySubscriptionNumber		2.75 ms	
searchPrepaidSubscriptionBySubscriptionNumber		2.65 ms	

YOUR RESULTS. YOUR WAY.

__SCAN Detection



Error Count

datafabric-prd-101	
__SCAN	8

datafabric-prd-102	
Query execution is too long	3
__SCAN	76

datafabric-prd-103	
__SCAN	73

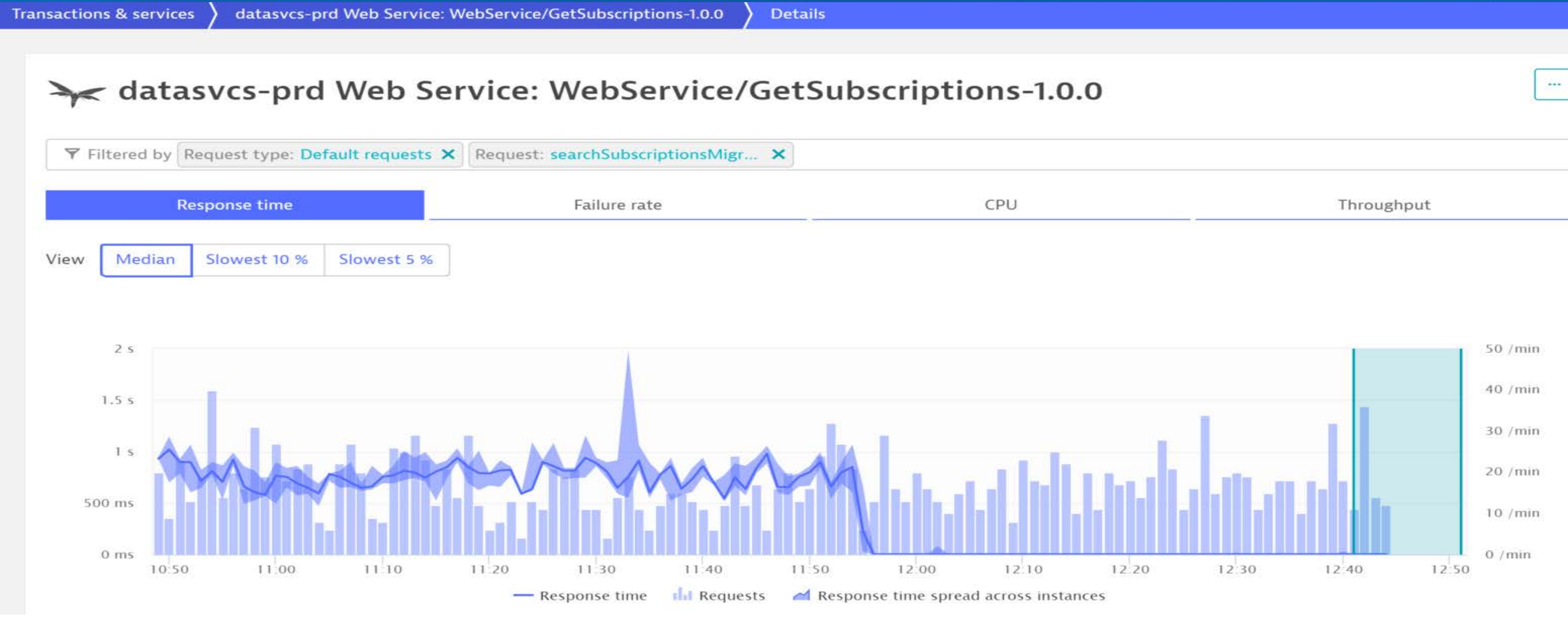
datafabric-prd-104	
Query execution is too long	1
__SCAN	76

datafabric-prd-105	
__SCAN	76

datafabric-prd-106	
Query execution is too long	4
__SCAN	76

datafabric-prd-107	
__SCAN	74

Tuning yields big results and stabilizes "The Grid"



Transactions & services 〉 datasvcs-prd Web Service: WebService/GetSubscriptions-1.0.0 〉 Details

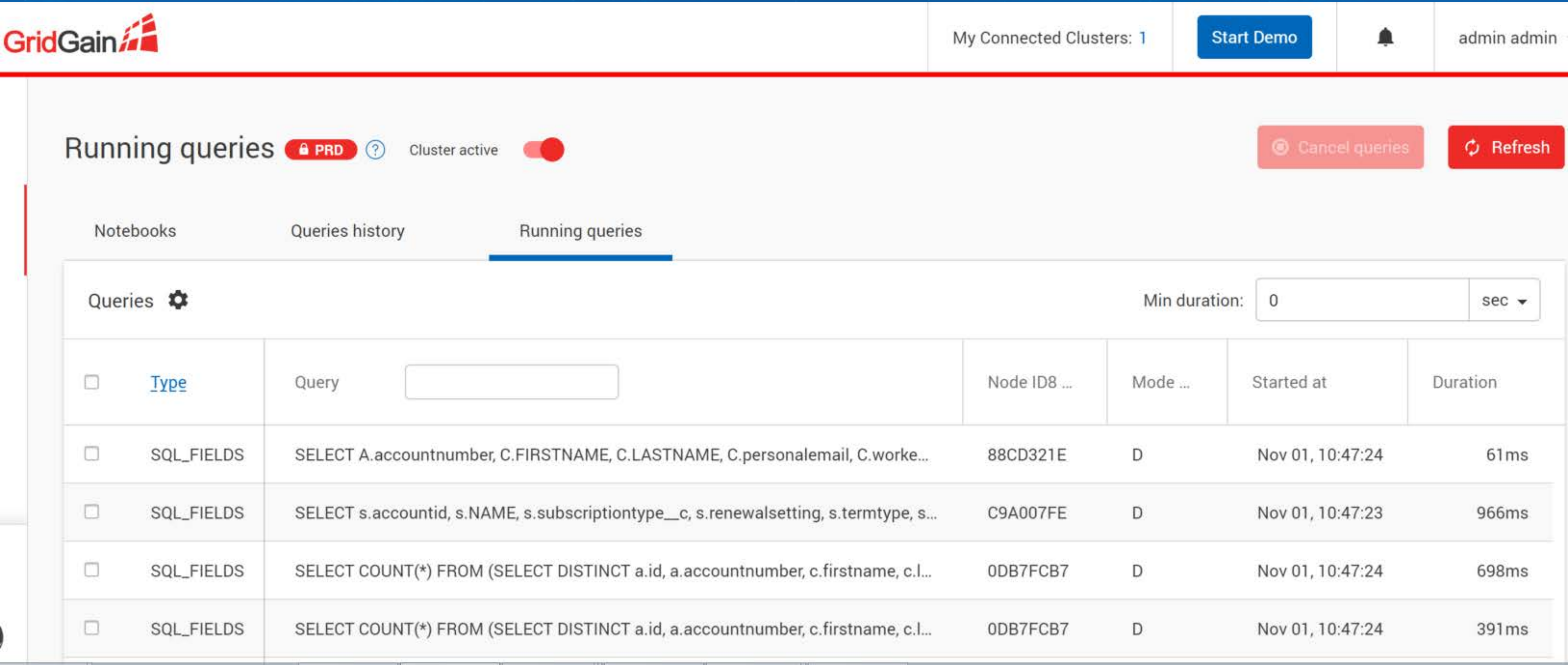
datasvcs-prd Web Service: WebService/GetSubscriptions-1.0.0

Filtered by Request type: Default requests ✕ Request: searchSubscriptionsMigr... ✕

YOUR RESULTS. YOUR WAY.

In-Memory Computing SUMMIT NORTH AMERICA 2019

"Long" Running Queries



GridGain

My Connected Clusters: 1 | Start Demo | admin admin

Running queries PRD Cluster active

Cancel queries | Refresh

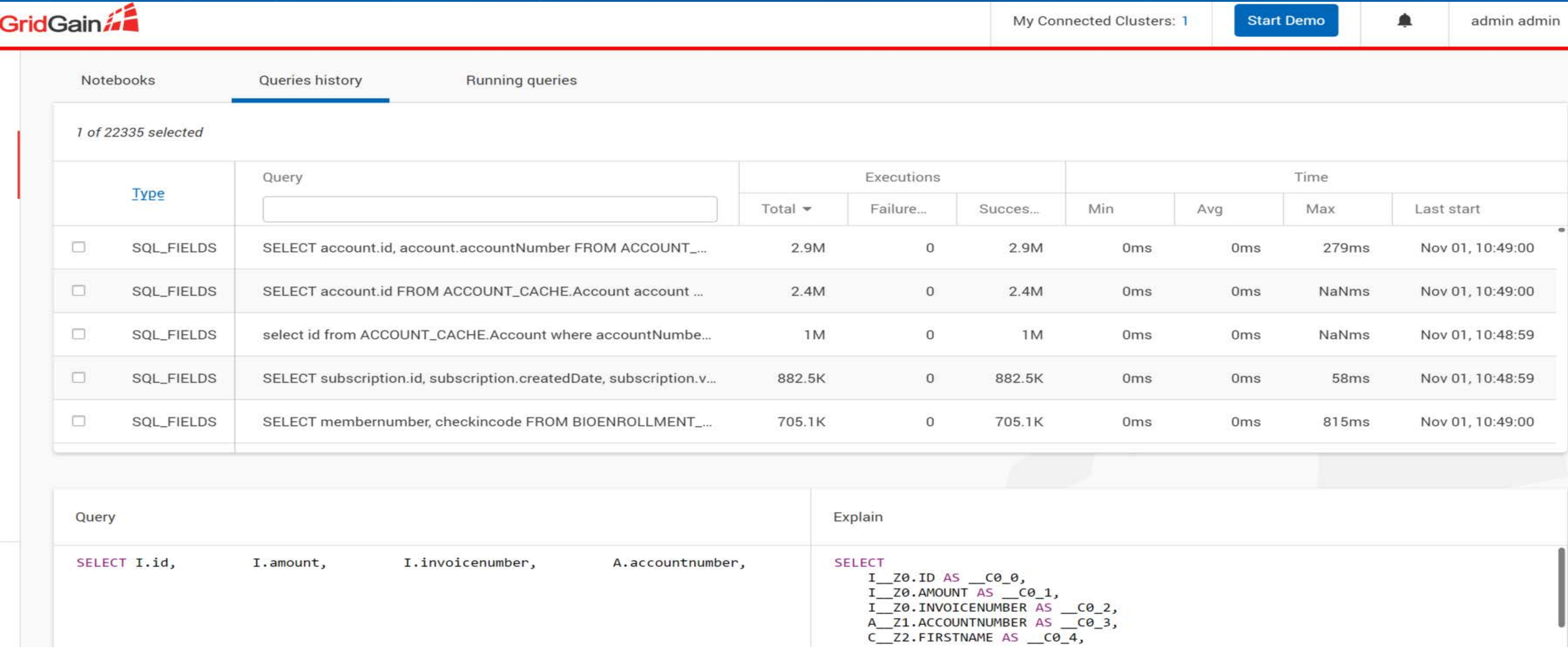
Notebooks | Queries history | Running queries

Queries

Min duration: 0 sec

	Type	Query	Node ID8 ...	Mode ...	Started at	Duration
☐	SQL_FIELDS	SELECT A.accountnumber, C.FIRSTNAME, C.LASTNAME, C.personalemail, C.worke...	88CD321E	D	Nov 01, 10:47:24	61ms
☐	SQL_FIELDS	SELECT s.accountid, s.NAME, s.subscriptiontype__c, s.renewalsetting, s.termtype, s...	C9A007FE	D	Nov 01, 10:47:23	966ms
☐	SQL_FIELDS	SELECT COUNT(*) FROM (SELECT DISTINCT a.id, a.accountnumber, c.firstname, c.l...	0DB7FCB7	D	Nov 01, 10:47:24	698ms
☐	SQL_FIELDS	SELECT COUNT(*) FROM (SELECT DISTINCT a.id, a.accountnumber, c.firstname, c.l...	0DB7FCB7	D	Nov 01, 10:47:24	391ms

Query History



GridGain | My Connected Clusters: 1 | Start Demo | admin admin

Notebooks | Queries history | Running queries

1 of 22335 selected

Type	Query	Executions			Time			
		Total	Failure...	Succes...	Min	Avg	Max	Last start
SQL_FIELDS	SELECT account.id, account.accountNumber FROM ACCOUNT_...	2.9M	0	2.9M	0ms	0ms	279ms	Nov 01, 10:49:00
SQL_FIELDS	SELECT account.id FROM ACCOUNT_CACHE.Account account ...	2.4M	0	2.4M	0ms	0ms	NaNms	Nov 01, 10:49:00
SQL_FIELDS	select id from ACCOUNT_CACHE.Account where accountNumbe...	1M	0	1M	0ms	0ms	NaNms	Nov 01, 10:48:59
SQL_FIELDS	SELECT subscription.id, subscription.createdDate, subscription.v...	882.5K	0	882.5K	0ms	0ms	58ms	Nov 01, 10:48:59
SQL_FIELDS	SELECT membernumber, checkincode FROM BIOENROLLMENT_...	705.1K	0	705.1K	0ms	0ms	815ms	Nov 01, 10:49:00

Query

```
SELECT I.id,          I.amount,          I.invoicenumber,          A.accountnumber,
```

Explain

```
SELECT
    I__Z0.ID AS __C0_0,
    I__Z0.AMOUNT AS __C0_1,
    I__Z0.INVOICENUMBER AS __C0_2,
    A__Z1.ACCOUNTNUMBER AS __C0_3,
    C__Z2.FIRSTNAME AS __C0_4,
```

YOUR RESULTS. YOUR WAY.

In-Memory Computing SUMMIT NORTH AMERICA 2019

Key Takeaways - Summary

Grid Gain can be use as a:

1. Key value object cache – jCache operations = fast data loading

2. SQL Queryable objects – runAttributesQuery = fast data access

3. System Of Record (SOR) – member club check in = file based database replacement

4. Solution that future proofs your company (more on that tomorrow)

YOUR RESULTS. YOUR WAY.™

Questions

Q&A

In-Memory Computing SUMMIT | NORTH AMERICA 2019

Appendix – Useful links

- My 2017 IMC Summit presentation adds context/details related to this presentation and our journey https://www.imcsummit.org/2017/us/sessions/how-in-memory-solutions-can-assist-saas-integrations
- Distributed joins, collocation, collocated queries and affinity https://apacheignite-sql.readme.io/docs/distributed-joins
- Don't over index: https://apacheignite.readme.io/v1.8/docs/indexes#indexes-tradeoffs
- GridGain Software Editions

  https://www.gridgain.com/products/software
- My 2019 IMC Summit Keynote presentation

https://www.imcsummit.org/2019/us/session/fitness-memory-computing-getting-ahead-game