In-Memory Computing SUMMIT EUROPE 2018

KAPPA, LAMBDA & MY JOURNEY FROM LEGACY TO NEW

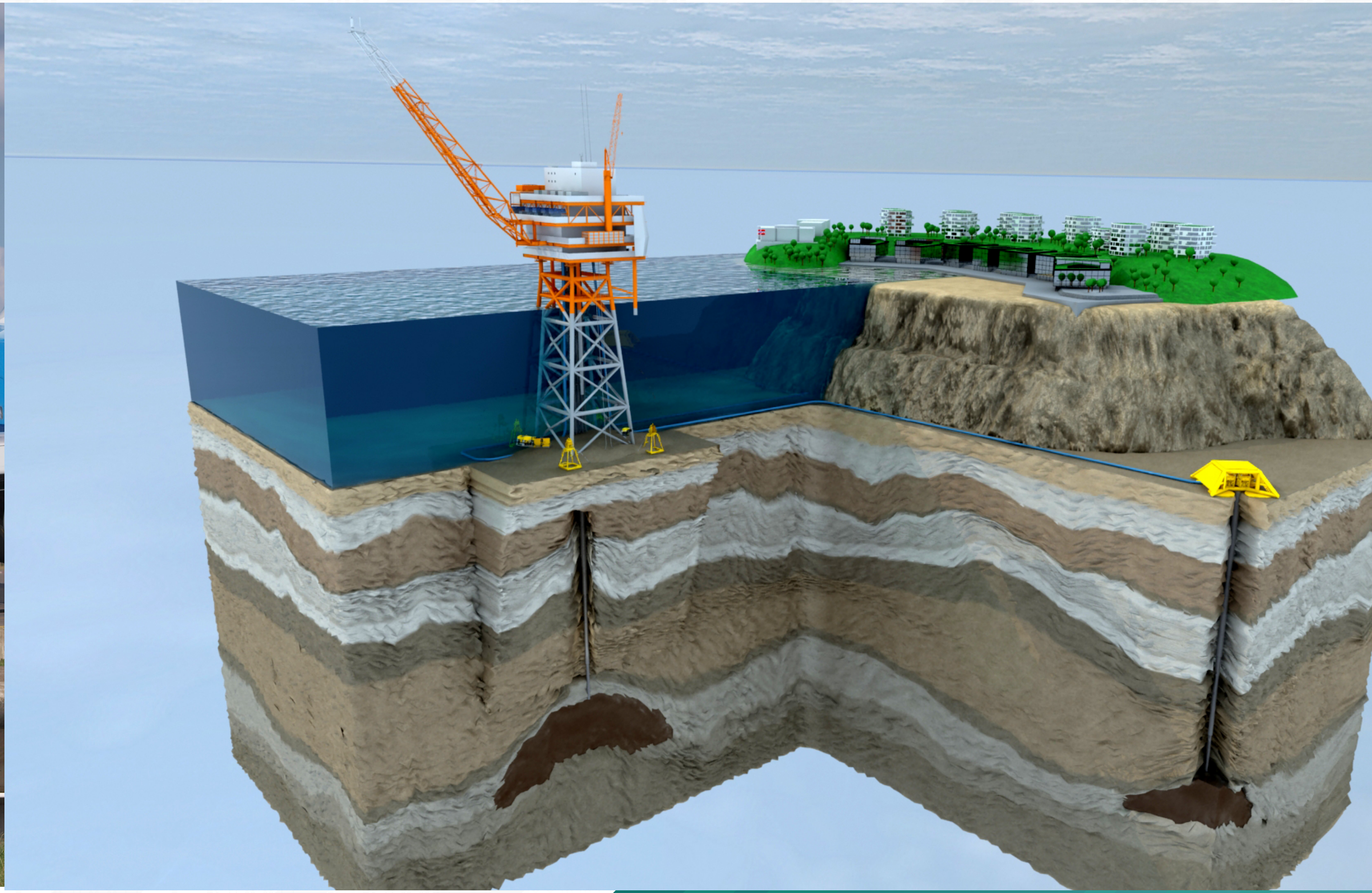
MICHAEL VAN DER HAVEN

CGI

OUTLINE

- This is not about…
- Exciting times (but they've always been!)
- The Legacy Bias
- The kind of stuff we develop
- The legacy we deal with
- Lambda & Kappa
- The next new (without killing off our legacy)

THIS IS NOT ABOUT….



TIMES ARE EXCITING!

1997 Throwback:

- In memory compute? You were king of the hill with a 64 MB PC
- Networking required 'Nuts & Bolts'
- The big divide: Concurrent computing / Grids / OpenMP & MPI only for research facilities and Fortune 1000 companies

2005 – Now:

- Internet → Cloud → Services → Cheap Data → Cheap processing → IoT → NoSQL → Data Lakes → Advanced Analytics → Machine Learning
- Open, Cheap and with the right credit card: available in a few hours or days

LEGACY BIAS

- We expect: Agility, Scalability, Cheap, Replacable, etc.
- Legacy Perception:

LEGACY

- Old
- Why did we ever build that?
- Hard to maintain
- Super heavy
- Monoliths
- $$$
- Etc.

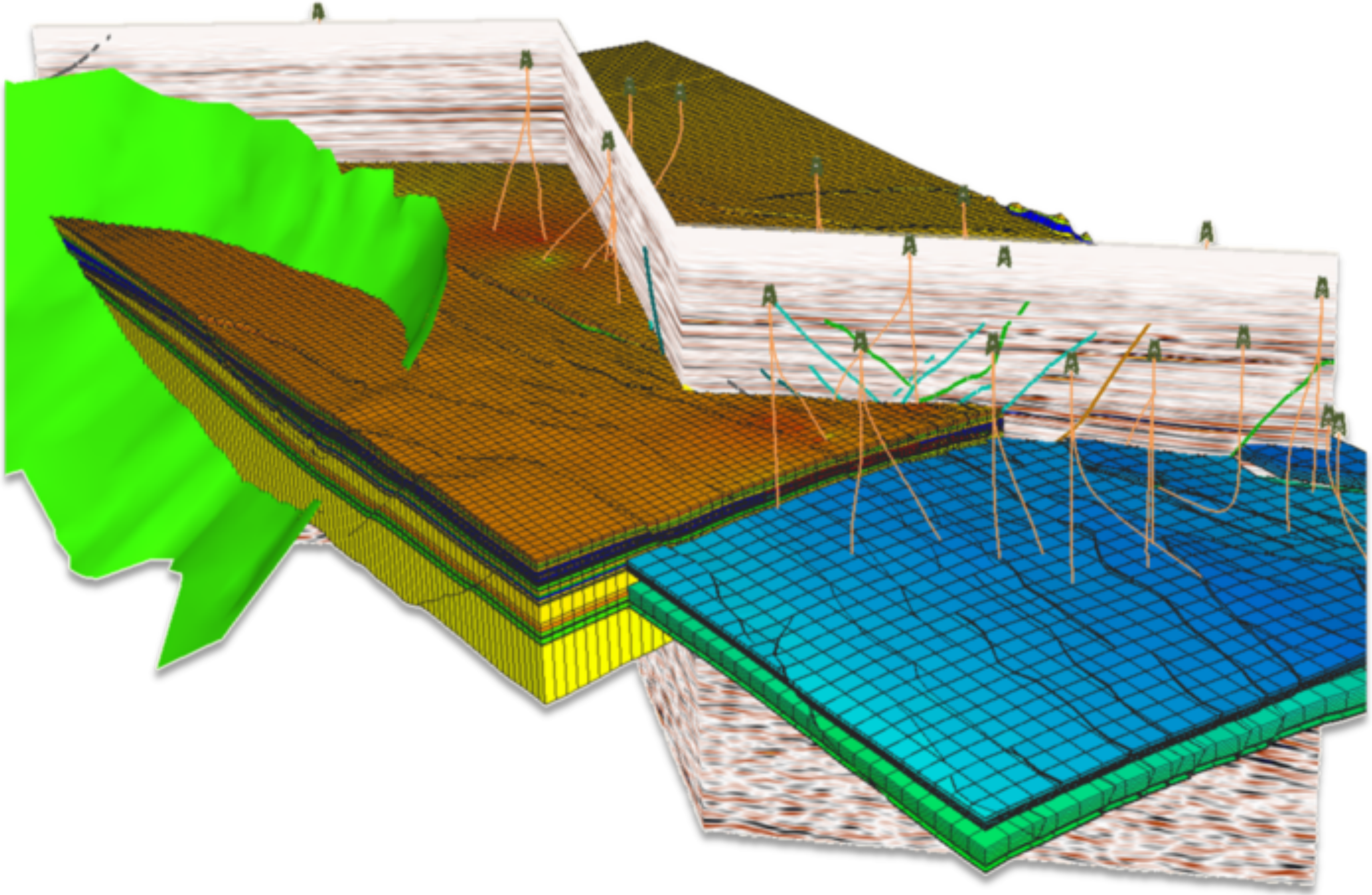
LIVE WITH IT

- $$$ spent with a reason
- Actively used to take $$$ business decisions (in our case: multi-billion $)
- Business Owners are happy enough

OR

- Not willing to spend $$$ on development again
- Etc.

WHAT MY TEAM BUILDS



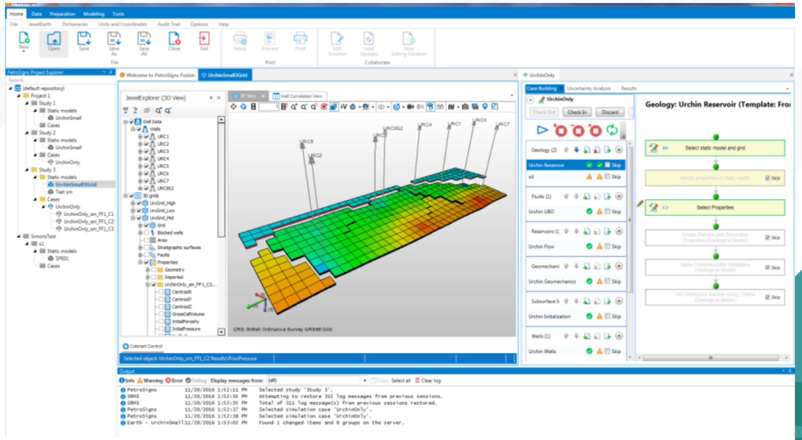
SUBSURFACE MODELLING AND OPTIMIZATION

- Collection of Disciplines that model
  - The layers in the ground
  - The faults and horizons
  - Structural Model
  - Physical and Chemical Rock Properties
  - Physical and Chemical Hydro Carbon Properties
  - Etc.

OUR LEGACY CHALLENGES

- We work together with our customer on building Modelling & Optimization platform Addressing these challenges:
  - Traditional separately operating disciplines
  - Work on one model → File hand over to next discipline
  - Separate tools
  - Big tools → 1 to 2 million lines of code each
  - Actively developed monoliths!
  - Brought to market by different vendors → limited control over implementation patterns
  - All data integration is ‘ingestion’ based

GOALS


Geology: Urchin Reservoir (Template: Fro
Select static model and grid
Select Properties
CRS: British Ordnance Survey GRS80 Grid
Colorset Control
Selected object: UrchinOnly_sm_FF1_C2 Results\PriorPressure
Output
Info
Warning
Error
Debug
Display messages from: (all)
Copy
Select all
Clear log
Urchin Initialization
Wells (1)
Urchin Wells
Skip

```
PetroSigns          11/20/2016 1:52:11 PM   Selected study 'Study 3'.
SRMS                11/20/2016 1:52:35 PM   Attempting to restore 315 log messages from previous sessions.
SRMS                11/20/2016 1:52:35 PM   Total of 311 log message(s) from previous sessions restored.
PetroSigns          11/20/2016 1:52:37 PM   Selected simulation case 'UrchinOnly'.
PetroSigns          11/20/2016 1:52:38 PM   Selected simulation case 'UrchinOnly'.
Earth - UrchinSmall11/20/2016 1:53:02 PM   Found 1 changed items and 0 groups on the server.
```

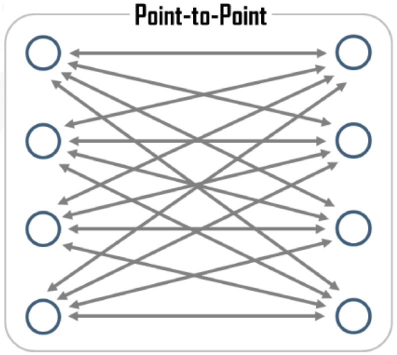
- No more files!
- Data at your finger-tips
- Single data view
- Each discipline can immediately cooperate with the other
- Single user experience
- Iterative modelling: Low Fidelity → Medium Fidelity → High Fidelity

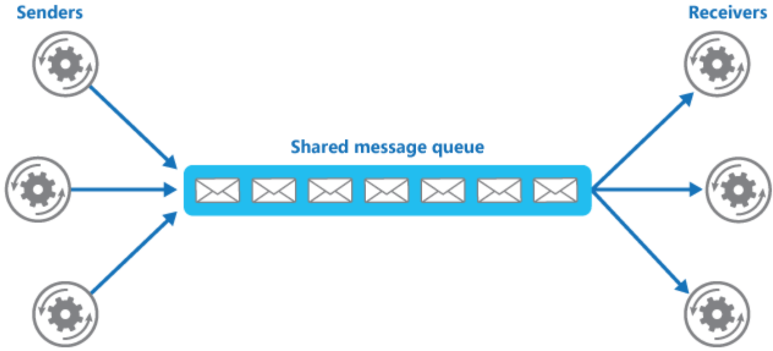
CHALLENGE: DATA, SIZE AND ACCESSIBILITY

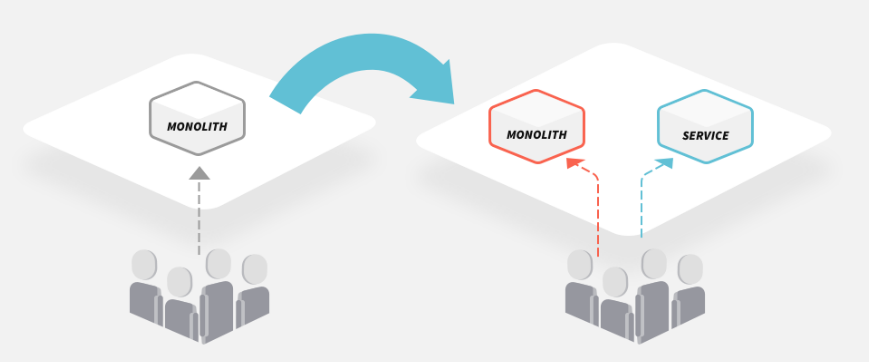
- Model Sizes run up to +/- 50 GB
- Real challenge is in ‘uncertainty’:
- Few thousand realizations per model
- ‘Traditionally’ not a problem: limited to the simulator that would throw away ‘unwanted results’
- Integrated tools that have ‘ingestion’ as main ‘implementation pattern’
- Data Explosion!

IMPLEMENTATION PATTERN

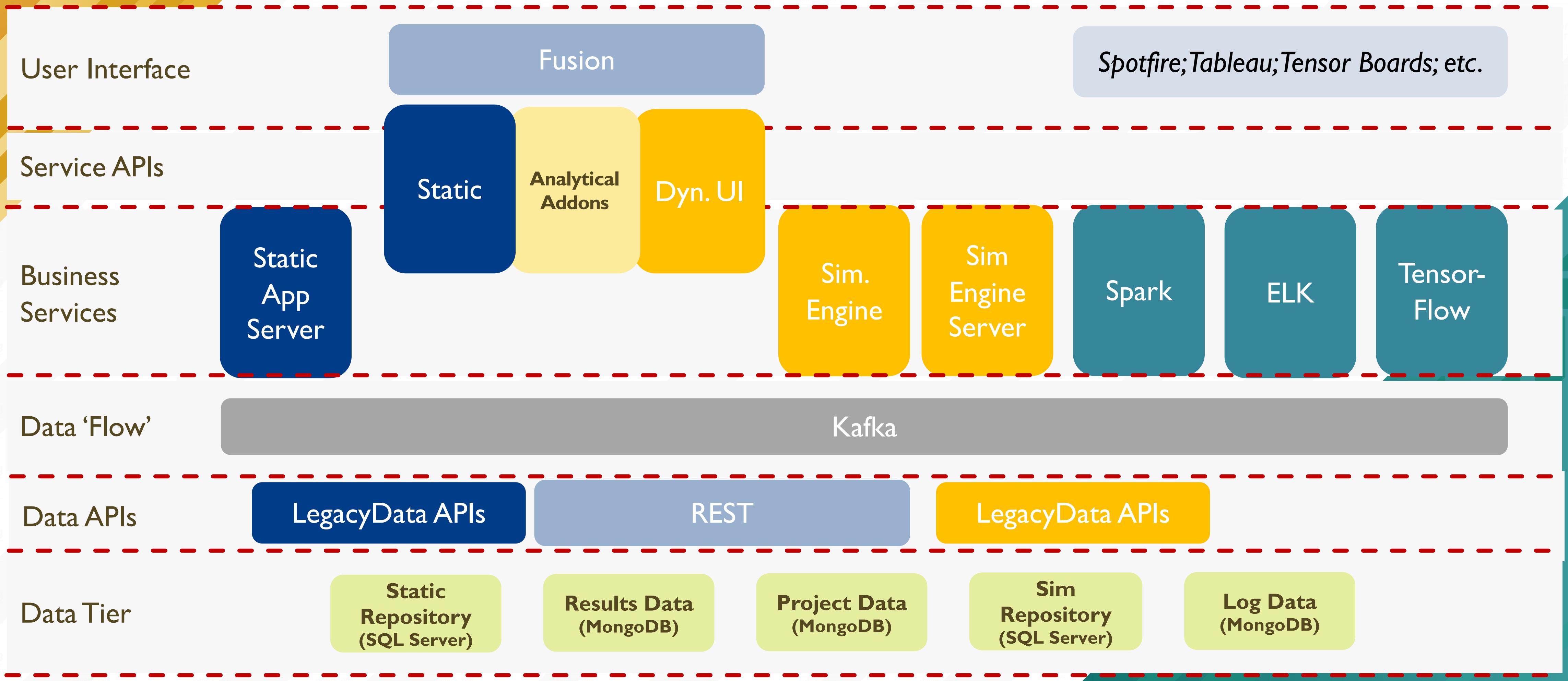
- Start out with connecting two applications
  - Early problem of PtP identified
  - Moved to Service Bus
- Monoliths 'Behave' like Service:
  - Introduce Edge API exposing services
  - Compute Monoliths in background containers
  - N.B. Has a notion of a 'wrangling' pattern, but unfortunately we do not have control over the vendor's tools







RESULT: CURRENT STATE OF AFFAIRS



User Interface

Fusion

Spotfire; Tableau; Tensor Boards; etc.

Service APIs

Static

Analytical Addons

Dyn. UI

Business Services

Static App Server

Sim. Engine

Sim Engine Server

Spark

ELK

Tensor-Flow

Data 'Flow'

Kafka

Data APIs

LegacyData APIs

REST

LegacyData APIs

Data Tier

Static Repository (SQL Server)

Results Data (MongoDB)

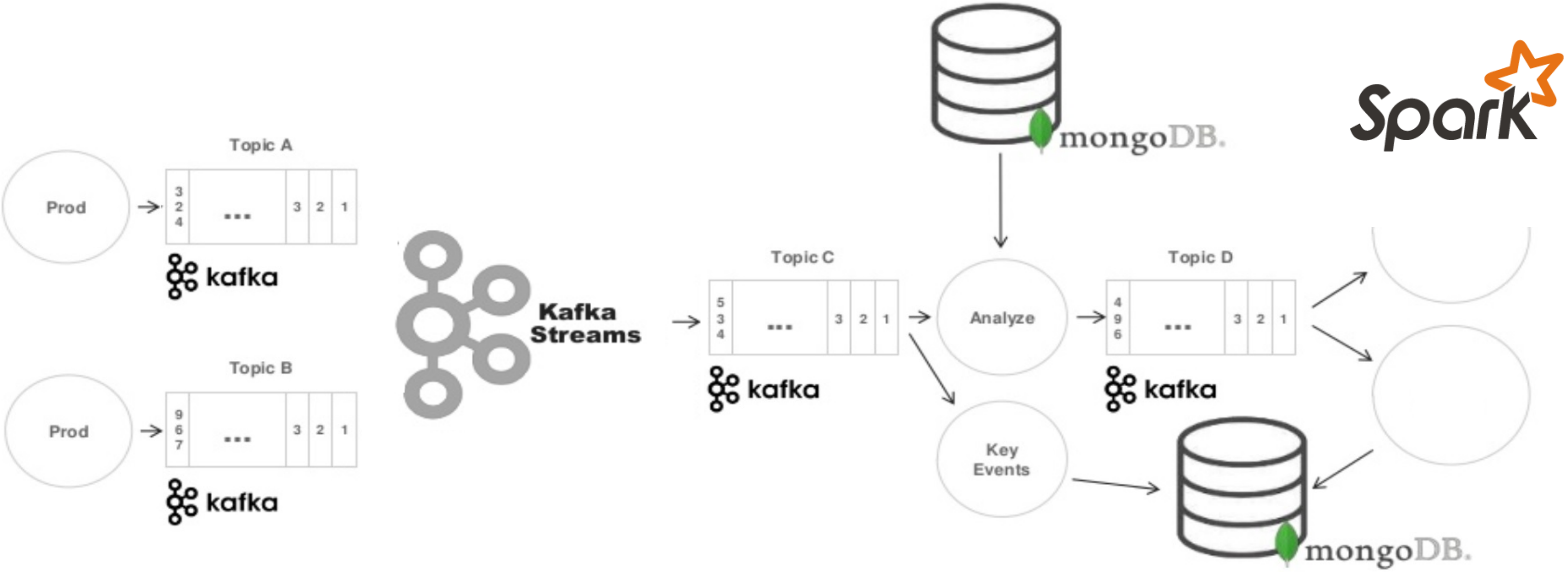
Project Data (MongoDB)

Sim Repository (SQL Server)

Log Data (MongoDB)

BUT WHERE IS THE IN-MEMORY PART?

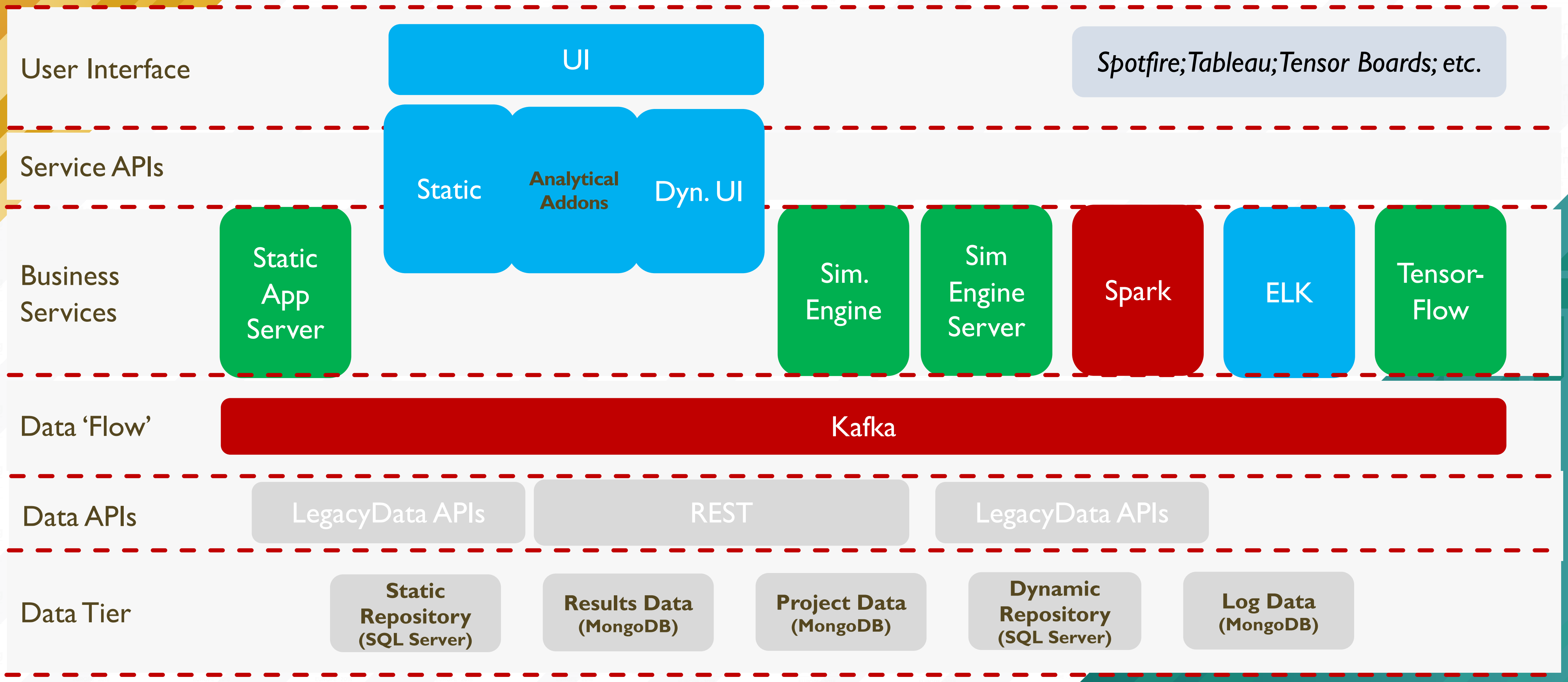
- The stream is our memory bus
- Spark is our 'Intelligent' Framework



LAMBDA? KAPPA?

- Lambda Architecture
  Three main layers:
  1. Speed
  2. Batch
  3. Serving

LAMBDA CHARACTERISTICS



User Interface

UI

Spotfire;Tableau;Tensor Boards; etc.

Service APIs

Static

Analytical Addons

Dyn. UI

Business Services

Static App Server

Sim. Engine

Sim Engine Server

Spark

ELK

Tensor-Flow

Data 'Flow'

Kafka

Data APIs

LegacyData APIs

REST

LegacyData APIs

Data Tier

Static Repository (SQL Server)

Results Data (MongoDB)

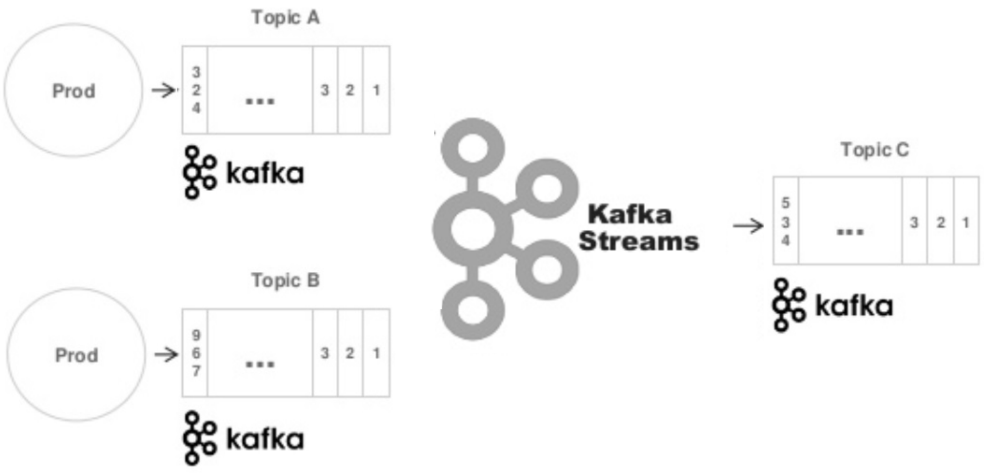
Project Data (MongoDB)

Dynamic Repository (SQL Server)

Log Data (MongoDB)

KAPPA?

- Kappa is (i.m.o.) about processing and creating results directly on the stream
  a. Instead of letting the stream be a carrier to fast & slow components and let them create results
  b. Process data on stream and let the results become a stream



UNCERTAINTY LEADS TO MORE KAPPA, LESS LAMBDA

- 80% of data produced is Simulated Results & Logs
- Simulated Results are created by uncertainty runs
- Choose Parameters (e.g. the length of a well perforation and the direction of rock permeability)
- Fill in a number of values with an uncertainty design (e.g. Monte Carlo / Box Bhenken / Etc.)
- Example: Parameter A has a distribution of 10 values, and parameter B has a distribution of 250 values
- Result: Table with values for column for Parameter A and Parameter B → 2,500 rows
- Each row is a simulation
- After running 2,500 simulations, determine which value had an impact (e.g. to match previous Quarter's production results)
- Normally: 25 values are picked as valid → Rest of data is thrown away (that's 120 TB 'temporal' data on average)

KAPPA ADVANTAGE

- Given: All Logs and Results are pushed onto Kafka
- Old Situation (Lambda):
  - Used to have connectors that ingest results into MongoDB
  - Then based on trigger → 25 cases are maintained, 2,475 cases are deleted from MongoDB
  - Software to write results
  - Software to read results
  - Software to delete results and 'earmark' results that need to be maintained, etc.
- New Situation (Kappa):
  - Results are on topic with retention time of few days
  - Trigger of 'maintain' cases is processed in KSQL and stored on new topic
  - Non-valid results are automatically discarded after retention period
  - Less Software (just KSQL) → Higher performance → Less maintenance (disk space)

BUT….. WHERE IS IGNITE?

- Not there yet, but…..

Scenarios:

- Legacy data components that require shared state for sessions
  - Built 8+ years ago in .Net
  - Can run on only one machine → Bottleneck → Ditch 'homegrown' state engine and replace by in-memory database
- One vendor uses SQL server to 'core dump' state after given events (e.g. time-steps)
  - SQL Server is 'misused' as ORM dump (every class has a table, no ref integrity)
  - Recognized that state is relevant for run-time and should be easily shareable among nodes (e.g. MPI context)
  - After 'run' state can be removed → Scalable in-memory database that accepts ORM

RECAP

- When we started out:
  Didn't think about Lambda or Kappa
- Queuing system was evaluated because we bumped our head on PtP
- Queuing is somehow hard for developers (it takes a while before a developer embraces queues over RPC)
  - Queuing had therefore potential impact Architecture & Developer attitude
- Think about Queuing
  - Usage Patterns (routed vs produced only / producer&consumer pattern / many consumers / etc.)
- Lambda started to emerge but adds complexity in number of components
- Kappa started to emerge, simpler but requires to be better aware of what you are doing (retention times need to be carefully chosen!)
- Lambda & Kappa live together!
  Lambda & Kappa are enablers and have achieved integration in a cost-effective manner (in fact, an integrated deployment turns out to be cheaper in run-time than the separate non-integrated tools)

MY JOURNEY TO THE NEXT NEW

- They say that the young can learn from the old and vice-versa, the same goes for IT
- Legacy is often a given, and even if you're on a migration path it can take a long time or is just too expensive
  (yes, sometimes you simply wait until that colleague reaches his pension)
- Try to embrace the new and incorporate into your project:
  - You don't always have to change jobs to work with cool new tech
  - Unless you have a boss that is in the 'I've been in this business 20+ years, and I know better'
- The $$$ spent (trust me, in our tools it is a lot of $$$) are not spent for nothing
  (modelling physics and chemistry is quite hard, and uncertainty doesn't make it easier)
- Our legacy has transformed into a bit of hype (e.g. Holistic Advanced Analytics and Machine Learning are all of a sudden possible)

THANK YOU!