PMDK Summit | Beijing | September 2019

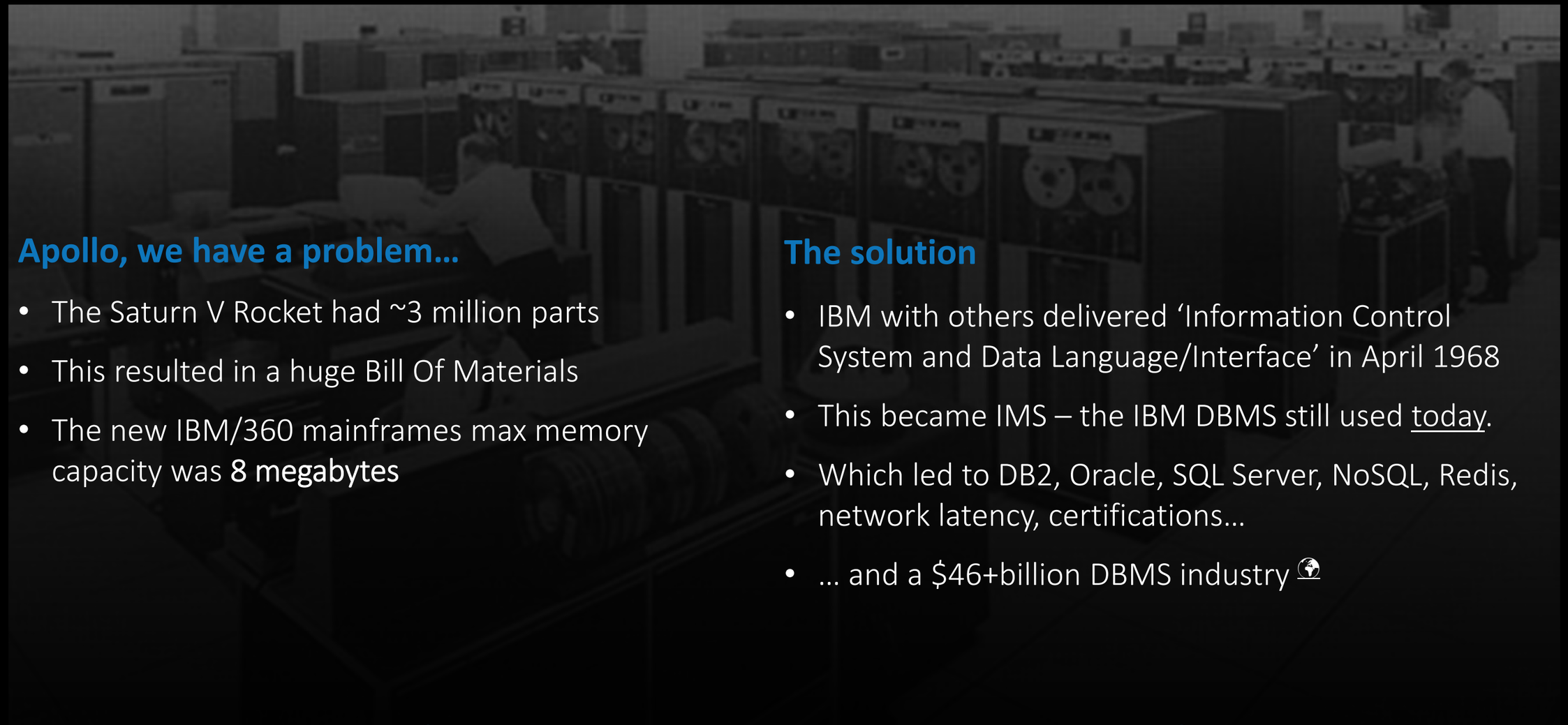
Introducing AxoMem

Sean Whiteley, Founder | axomem.io

axomem

51 YEARS AGO

The Apollo program led to the IT doctrine of application & data separation.



Apollo, we have a problem…

- The Saturn V Rocket had ~3 million parts
- This resulted in a huge Bill Of Materials
- The new IBM/360 mainframes max memory capacity was **8 megabytes**

The solution

- IBM with others delivered ‘Information Control System and Data Language/Interface’ in April 1968
- This became IMS – the IBM DBMS still used today.
- Which led to DB2, Oracle, SQL Server, NoSQL, Redis, network latency, certifications…
- … and a $46+billion DBMS industry

Separation of application and data was borne of necessity!

APRIL 2019…

Intel® Optane™ DC Persistent Memory launched

Persistent Memory changes things

- More than ample memory (in both capacity and latency) for a wide range of applications.
- PMDK provides a suite of libraries for developers to interface to Persistent memory.
- Its also tackling topics that were traditionally DBMS (transactions, replication, etc).

But we have no clear precedents…

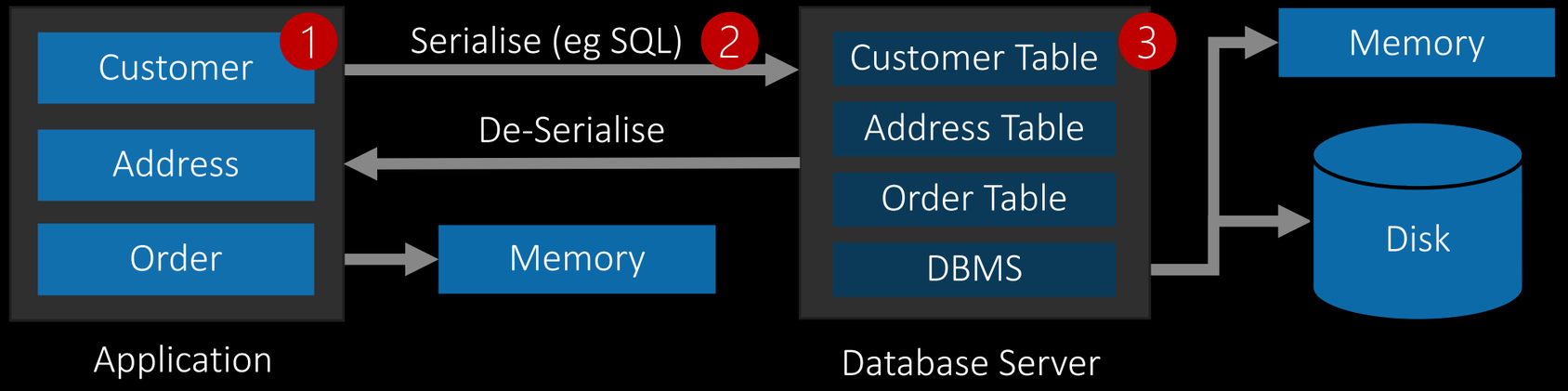
- How build a modern business application that only requires CPU and Memory?
- How to architect an x86 application with high multiprocessing and direct access to 24TB memory?
- How to efficiently integrate with traditional IT systems?
- How to monitor, back up/restore, patch and upgrade those applications (the operations aspects)?

How do we define a “Persistent Application”?

PERSISTENT APPLICATIONS

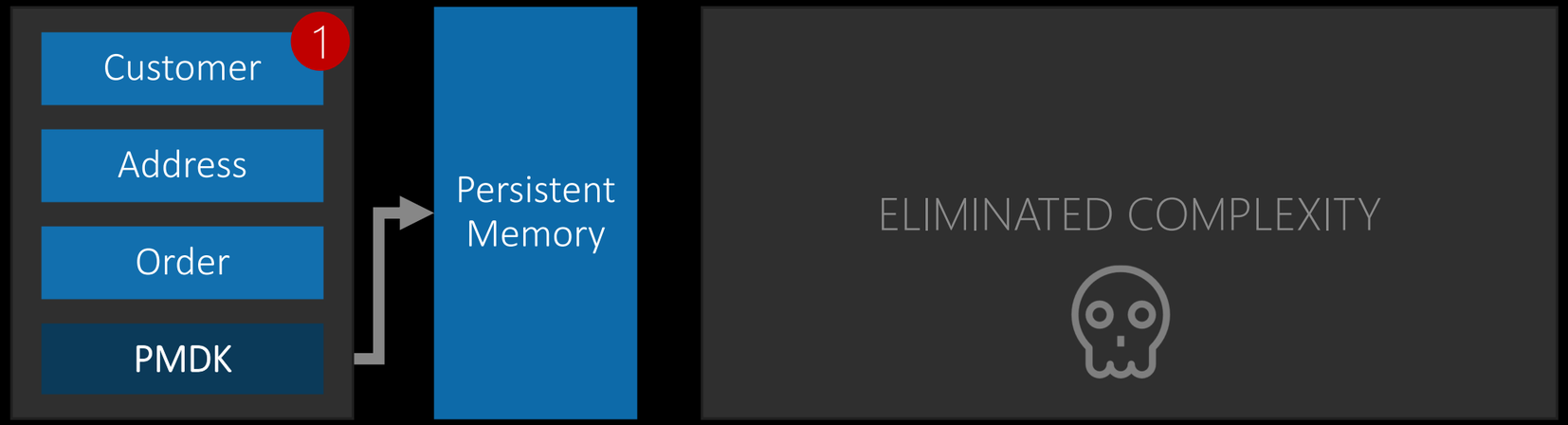
Persistent Memory and PMDK can radically simplify data management complexity in applications

TRADITIONAL ARCHITECTURE



- Requires at least three “variations” of business object representation
- Need more domains skills
- Boundaries slow development & runtime speed

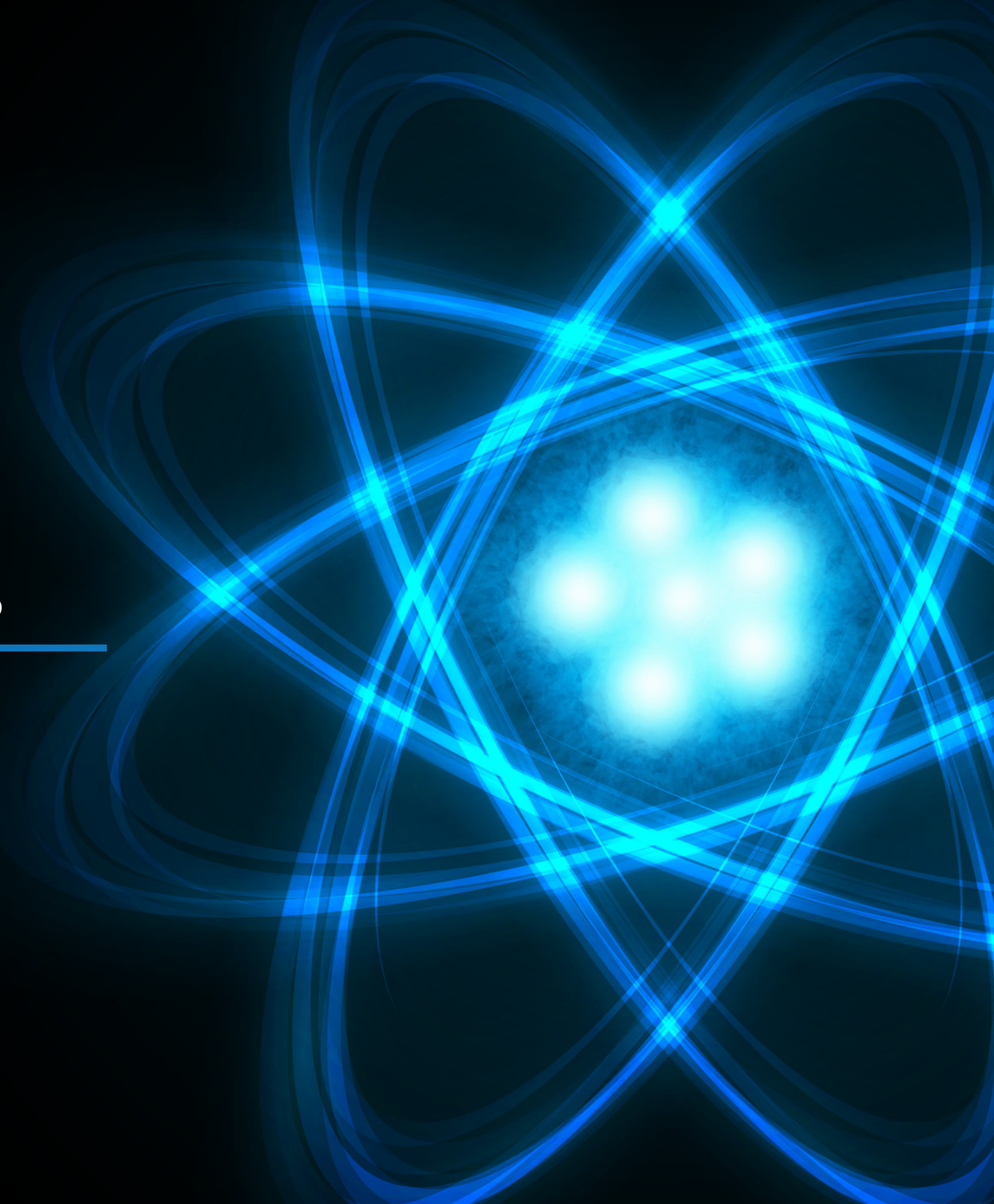
PERSISTENT APPLICATION



- Only one instance of each business object
- All objects are available all the time.
- Faster development and runtime speed

Announcing AxoMem Nucleus

An open source framework for Persistent Applications



AXOMEM NUCLEUS ← *Announced here at PMDK Summit!*

Nucleus is an Open Source accelerator for building native Persistent Memory applications

Simple framework for building apps

Allows developers to start building their own business apps while Nucleus handles common functions.

Built in ReST Server

Extremely fast ReST-based server and easy-to-use JSON library. Developers can easily add their own routes.

Use C++ variables and containers

Developers can use any of the PMDK-supported C++ types and not have to worry about serialisation.

Built in Configuration and Logging

Conf/ini based configuration, and flexible logging engine pre-integrated.

Cross Platform support

Tested on Linux (Fedora v29+) and Windows 10 (VS 2017 v15), using cmake build system.

Open Source

High flexibility for re-use and incorporating 3rd party contributions and modules.

EXAMPLE CODE

Example of using native LibPMemObj-C++ semantics and adding a ReST API

Main.cpp

```
int main(int argc, char *argv[]) {

    return nucleus::Nucleus<MyApp>(argc, argv).Run();

};
```

MyApp Header

```
class MyApp {

public:
    MyApp();            // this at pool creation or app reset. It does…
    ~MyApp();           // this happens when the class instance is being…

    void Initialize();  // this happens at object creation, typically…
    void Start();       // this happens each time the applications runs

private:
    // These are the persistent memory objects for this class.
    persistent_ptr<string> p_message;
    p<int> p_update_count;
};
```

MyApp Constructor

```
MyApp::MyApp()
    : p_message{make_persistent<experimental::string>("Hello World")}
    , p_update_count{0}
{
    Logging::log()->debug("MyApp Persistent Constructor called");
}
```

Add ReST API - GET

```
router->http_get(
    R"(/api/v1/app/message)",
    [&](auto req, auto params) {
        json j = "{}"_json;

        j["data"]["value"] = p_message->c_str();

        return req->create_response()
                .set_body( j.dump())
                .done();
    });
```

Add a ReST API - UPDATE

```
router->http_put(
    R"(/api/v1/app/message)",
    [&](auto req, auto params) {

        auto j_req = json::parse(req->body());
        std::string message_value = j_req["value"];
        Logging::log()->trace("MyApp Message is being set to {}.", message_value);

        pmem::obj::transaction::run(
            PoolManager::getPoolManager().getPoolForTransaction(), [&] {
                p_message->assign(message_value);
                p_update_count++;
        });

        json j = "{}"_json;
        j["response"]["message"] = fmt::format("Message value updated {} time(s)
                                                so far", p_update_count);

        return req->create_response()
                .set_body( j.dump())
                .done();
    });
```

BUILD AND RUN NUCLEUS

Example of using native LibPMemObj-C++ semantics and adding a ReST API

Building

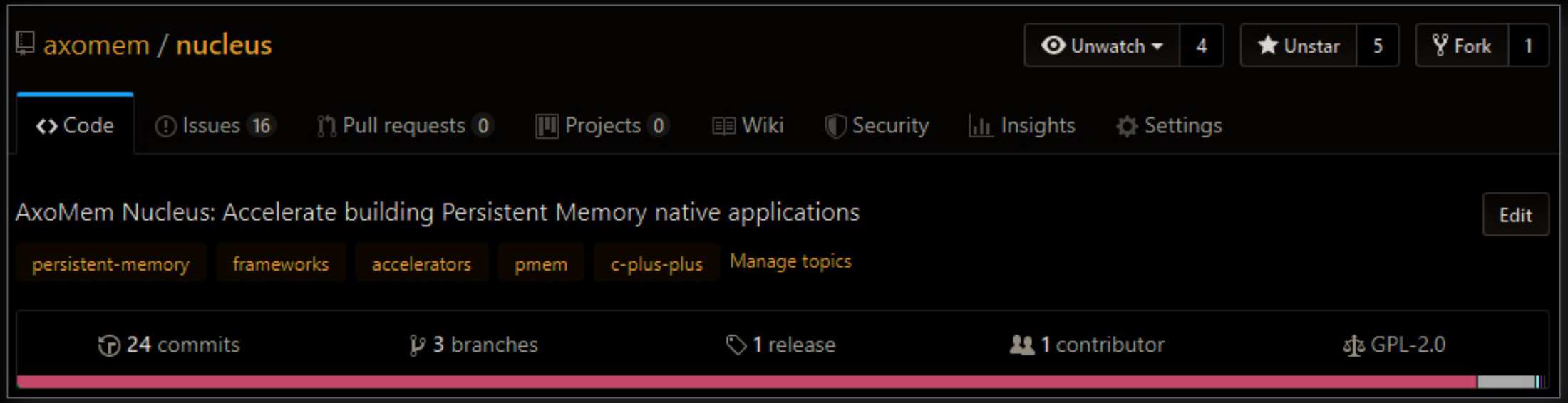
```
[sean@localhost build]$ cd bin
[sean@localhost bin]$ ./nucleus
[2019-09-05 19:06:05.872] [main] [info] Logging has been initialised with name main and loglevel debug and
saved to ./nucleus.log
[2019-09-05 19:06:05.872] [main] [info] Built with Nucleus. See https://axomem.io for more info, and follow
 us on Twitter @axomemio for updates
[2019-09-05 19:06:05.872] [main] [info] Nucleus is starting
[2019-09-05 19:06:05.872] [main] [debug] Creating PoolManager with pool file ./nucleus.pmem
[2019-09-05 19:06:05.872] [main] [info] Creating new pool ./nucleus.pmem with layout 'myapp_v0.0.1' and siz
e 1073741824
[2019-09-05 19:06:05.929] [main] [info] Pool successfully created.
[2019-09-05 19:06:05.929] [main] [warning] Please remember Nucleus is alpha. Things *will* change and there
 *will* be bugs!
[2019-09-05 19:06:05.933] [main] [debug] MyApp persistent object not yet initialized - persisting MyApp Obj
ect
[2019-09-05 19:06:05.933] [main] [debug] MyApp Persistent Constructor called
[2019-09-05 19:06:05.935] [main] [debug] AppManager is opening Application
[2019-09-05 19:06:05.935] [main] [debug] Current App State is NEW
[2019-09-05 19:06:05.935] [main] [info] AppManager: App initializing after first persistence
[2019-09-05 19:06:05.937] [main] [debug] MyApp is starting
[2019-09-05 19:06:05.939] [main] [debug] AppManager is creating ReST Server
[2019-09-05 19:06:05.939] [main] [info] ReST Server configured on port 8080 with 4 threads across 96 CPUs
[2019-09-05 19:06:05.940] [main] [debug] AppManager Entering Main thread run loop with App State RUNNING

***
Nucleus is running. Default site is http://localhost:8080/api/v1/ready
Press CTRL-C once to shutdown normally. May require up to 3 presses in abnormal termination
***
```

Running

```
[sean@localhost ~]$ curl http://127.0.0.1:8080/api/v1/app/message -w "\n\n"
{"data":{"value":"Hello World"}}

[sean@localhost ~]$ curl -i -X PUT -d '{"value":"Hello PRC"}' http://127.0.0.1:8080/api/v1/app/message -w "
\n\n"
HTTP/1.1 200 OK
Connection: keep-alive
Content-Length: 65
Access-Control-Allow-Origin: *

{"response":{"message":"Message value updated 1 time(s) so far"}}

[sean@localhost ~]$ curl http://127.0.0.1:8080/api/v1/app/message -w "\n\n"
{"data":{"value":"Hello PRC"}}

[sean@localhost ~]$
```

AVAILABLE NOW ON GITHUB

Nucleus released to the community to start engagement on Persistent App development



https://github.com/axomem/nucleus

Current Features

- Easy to get example MyApp running – clone, cmake then make (hopefully)
- Easy to expand – add more classes, add ReST end Points
- Supports Linux and Windows

Near-term targets

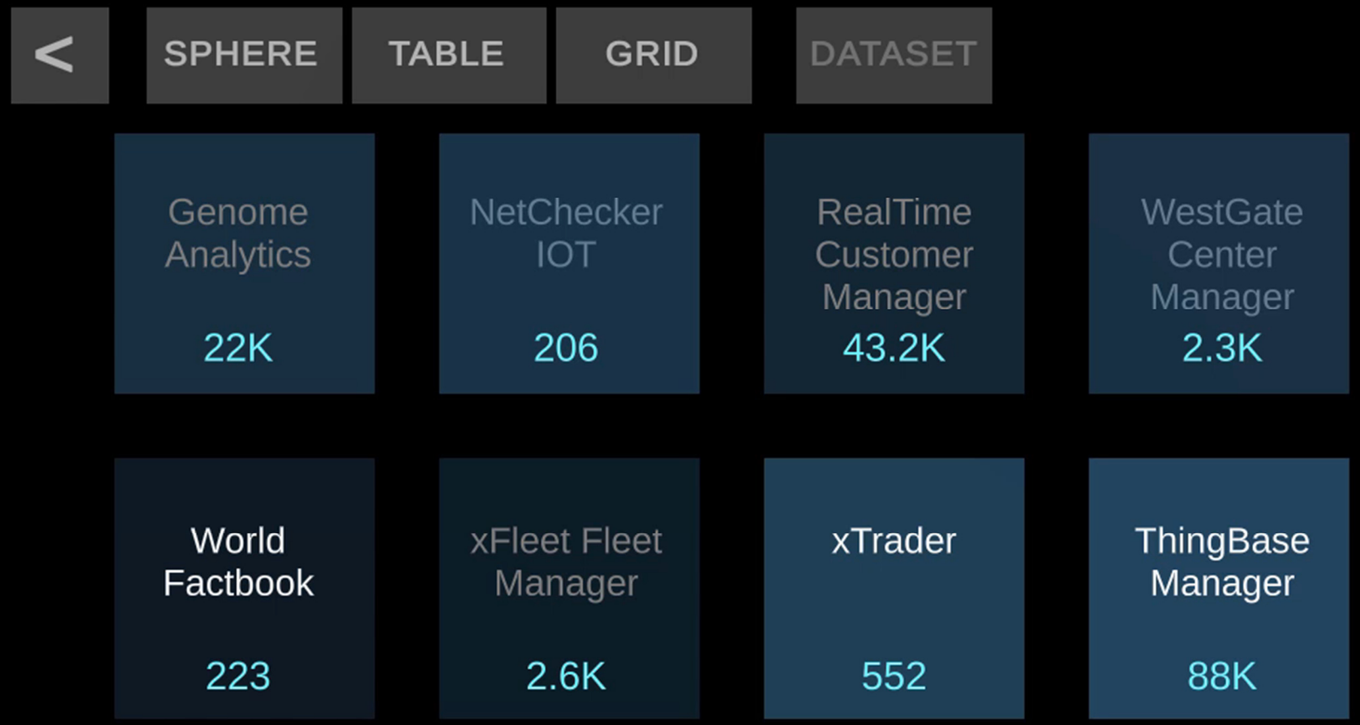
- Launch at PMDK Summit
- Increase usage and get feedback from developers
- Expand examples, test framework, benchmarks

AxoMem ThingBase & xScape

Build data-oriented applications that support AI, Automation & Visualisation

axomem

DEMO – THINGBASE & XSCAPE



https://youtu.be/d2z_ytTySJw

AxoMem xScape UI Demo

INTRODUCING THINGBASE AND XSCAPE

Analyze, visualize and automate in large-scale data environments

End-user oriented data visualisation

Native 3D UI to visualize complex relationships supporting multiple display devices including VR & AR

Runtime Analytics and AI execution

Real-time or scheduled execution of algorithms driving enterprise automation based on ML and AI

High capacity and throughput

Supports hundreds of millions of operations per second and scaling from mobile-scale to multi-terabyte scale

Consistent data interface

Hosted algorithms consume data from disparate sources using one data interface

Persistent Memory Support

Capable of supporting 24TB on a single server, or scaling horizontally to hundreds of TB at memory-like speed

IOT & Graph oriented

Billions of records in flexible data model and supporting large time-series structures

Note: AxoMem products are pre-beta and features may change

INSIGHT FROM VISUALISATION

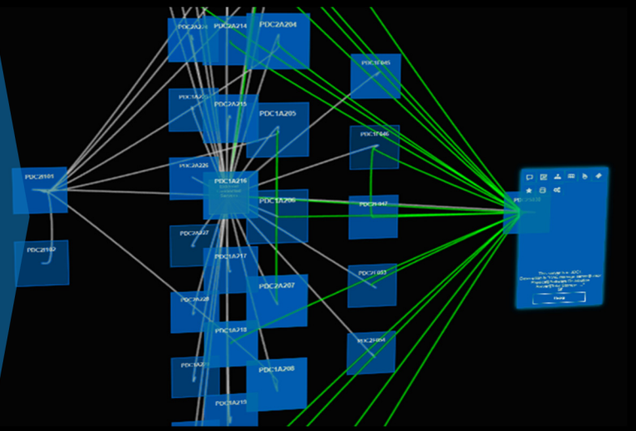
Engaging visualisations from high-level dashboards to detailed views with fast, continuous navigation

Top-level dashboards

Aggregate data for higher-level monitoring and insight. Drill down to get more data

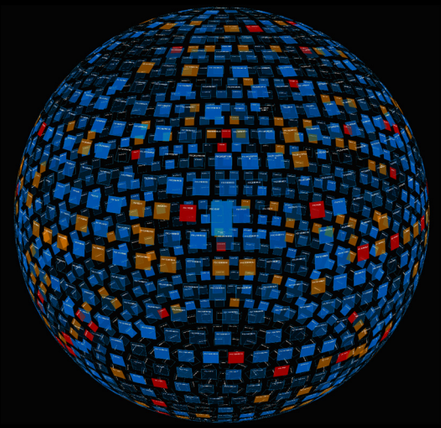
Relationships from live data

See and understand complex relationships and patterns derived live from your data, modelled in 2D or 3D



Large-scale data views

Get the big picture from large datasets presented in layouts driven by algorithms



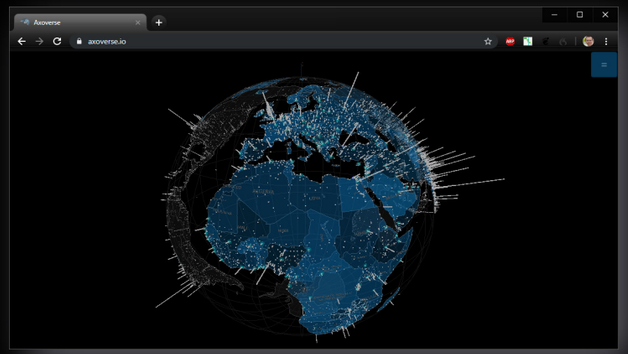
Detailed statistics

Drill down to see detailed graphs and tables. Add data elements as needed to discover root causes.

* Examples from previous client engagements.

ENGAGING INTERACTION

Targeting devices from browsers through to interactive room-level displays



Web Browser

VR/AR Capable

Tablet

High-Spec Workstation[1]

Wall Touch Panel & Digital Signage

Room-Level Displays

DEMO – XSCAPE AUGMENTED REALITY

axomem

Pret > Bil

https://youtu.be/i1JZiw8_uKU

AxoMem xScape Augmented Reality Demo

PERFORMANCE SNAPSHOT – 2 SOCKET SERVER

23 billion trades loaded for 555 S&P500 securities. Loading speed 2m+ records/sec. **Reading 1+ billion /sec**

Data Characteristics

- Loading trades for 555 S&P500 companies since 2009
- 23 billion trades in 555 separate CSV files
  - Average per security = 41m trades
  - Max is 357 million trades (Bank of America)
- Persistent Memory Pool split across two regions (sockets) in AppDirect mode, using FS-DAX on ext4 filesystem
- Trades loaded into single PMemObj array per company
- All companies and trades in the same memory space

Server Characteristics

- 2x next gen Intel® Xeon® Scalable processors (Cascade Lake)
- 3TB total Intel® Optane™ DC Persistent Memory
- 768GB RAM, 1.2 TB SSD

Data Ingestion

- Ingesting ~2m+ records per second from CSV
- Every row is a persistent memory transaction commit
- Multiple CSV Import optimizations for speed

Data Reading - example

- Calculating average trade size across companies
- Single threaded = 40m reads /sec
- Multithreading reaches over 1 billion reads / sec
- Not NUMA optimized yet – more gains expected

Restart Time

- Restart time less than 500 milliseconds to shutdown + restart idle application

PERFORMANCE SNAPSHOT – ANALYTICS (READ)

Example read rate for averaging calculations across 20 billion trades, 4 data elements per trade

https://youtu.be/eZc6B95-GD0

AxoMem Products

NUCLEUS

Framework for building Persistent Memory Applications.

https://github.com/axomem/nucleus

THINGBASE

Fast, flexible platform for building dynamic datascapes including analytics, AI and automation.

XSCAPE

Cross-platform app enabling fluid 3D web, VR and AR visualization of small or large datascapes.

Note: AxoMem products are pre-beta and features may change

PMDK SUMMIT PRC DEMOS AND HANDS-ON

Drop by and see the demos, and get started on PMDK development in the Hands-on session

```
(/api/v1/app/message)",
](auto req, auto params) {

  auto j_req = json::parse(req->body());
  std::string message_value = j_req["value"];
  Logging::log()->trace("MyApp Message is being set to {

  pmem::obj::transaction::run(
      PoolManager::getPoolManager().getPoolForTransactio
          p_message->assign(message_value);
          p_update_count++;
  });

  json j = "{}"_json;
  j["response"]["message"] = fmt::format("Message value
```

Touch-screen Demo

ThingBase running on a touchscreen in the main hall*

VR Demo

ThingBase running on Samsung S8 & Oculus Gear VR

Hands On Lab

Join the Hands-on session to learn more about PMDK

CONTACT AXOMEM

Let’s start a conversation!

谢谢 - THANK YOU!

Address

AxoMem Pte Ltd
14 Robinson Road #08-01A
Singapore 048545, Singapore

Phone & Email

Direct Line: +65 3138 4141
info@axomem.io

Links

https://axomem.io/
twitter.com/axomemio
linkedin.com/company/axomem
https://github.com/axomem