Parallel patterns against an exabyte data lake using exascale heterogeneous computing

Mr Andreas Vermeulen
University of St Andrews
Saint Andrews, Fife KY16 9AJ
University of Dundee
Nethergate,Dundee DD1 4HN
a.f.vermeulen@dundee.ac.uk

Dr Vladimir Janjic
University of St Andrews
Saint Andrews, Fife KY16 9AJ
vj32@st-andrews.ac.uk

Mr Andy Cobley
University of Dundee
Nethergate, Dundee DD1 4HN
acobley@computing.dundee.ac.uk

ABSTRACT

An enhancement of a rapid information factory using exascale heterogeneous computing and parallel knowledge-extraction patterns to generate a deep learning source from a exabyte data lake.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous

General Terms

Theory, Framework, Application, Research, Hardware

Keywords

exabyte, exascale, knowledge-extraction, patterns, rapid information factory, RIF, RIFF, RIFC, heterogeneous computing, parallel patterns, mapr, cassandra, spark, opencl, R, fastflow, cuda, 3D torus network, deep learning, machine learning

1. RESEARCH QUESTION

Can a Rapid Information Factory using agile and lean six sigma manufactory principles to solve the issues generated by effective and efficient exascale heterogeneous computing of a quintillion bytes data lake into a value-add deep learning knowledge source?

2. RAPID INFORMATION FACTORY (RIF)

2.1 Rapid Information Factory Framework (RIFF)

The rapid information factory framework is a methodology, the result of research since 2008 , designed to guide a exascale [3] heterogeneous computing cluster to process a exabyte data lake. The framework will processes a quintillion calculations per second against quintillion bytes of disk storage. The framework generates a series of virtual factories that together process the data lake using enhanced custom designed parallel processes.

2.1.1 Functional Layer

The functional layer handles the all functional processes within the cluster. The functions is build to empower the factory to process data sources in a predictable and repeatable series of processes. This layer is the bulk of the framework, as it contains the main components of the factory process and will expand as the factory deploys into full production.

High-Level View

The high-level view of the Homogeneous Ontology for Recursive Uniform Schema (HORUS) shows the users the current status of the rapid information factory. This is achieve by visualisation of the rapid information factory via a Rstudio Shiny [12] and R [17] based web site. The complete state of the factories are online in a graph database.

Synaptic Assimilator (SA)

The synaptic assimilator (SA) is an artificial intelligence [11] engine that performs the processes assigned to the system to the most effective and efficient method.

The artificial intelligence uses machine learning as an investigation and testing method to improve and select the correct combination of processing artifacts to achieve the efficient and effective outcome of each factory process.

The engine is taught data science and performance improvement processes to enables it to manage the factories to process the data sources it is supplied with for processing.

Exascale Data Lake

The exascale data lake is a data source that exceeds a quintillion bytes. The data source holds structured, semi-structured and unstructured data. The synaptic assimilator converts it into a base deep learning data source by applying the appropriate functions and data processing patterns.

Persistent Recursive Information Schema Manipulator (PRISM)

The persistent recursive information schema manipulator is the central control framework for each data procesing flow through the system using a bulk synchronous parallel (BSP) abstract computer as a bridging model for rapid information factory's parallel algorithms, that are pre-defined and tested by each factory.

RAPTOR Supersteps

The RAPTOR framework is the basis for the six supersteps of the bulk synchronous parallel (BSP) based process. The framework uses fundamental building blocks like pipeline, farm and loopback to formulate more complex structures to handle the data requirements within eeach of the six supersteps.

The six supersteps are:

Retrieve Superstep

The retrieve superstep is responsible for data retrieval from other data sources into the data lake. (See Retrieve Superstep for more details)

Assess Superstep

The assess superstep is responsible for the data validation in the active factory. Data Quality is determined and also improved where possible in this super step. (See Assess Superstep for more details)

Process Superstep

The process superstep is responsible for the processing of the dark data in the data lake into a structured data vault. The data vault keeps full record of the different phases that the data is process over time. (See Process Superstep for more details)

Transform Superstep

The transform superstep is responsible for transforming the data vault into an enterprise data warehouse. This superstep tranforms the data into knowledge by adding dimensionality and insight to the data vault. (See Transform Superstep for more details)

Organise Superstep

The organise superstep is responsible for organising the data sets together for each business grouping from the data warehose into data marts. (See Organise Superstep for more details)

Report Superstep

The report superstop is responsible to perform the required reporting and analytic requirements. (See Report Superstep for more details)

2.1.2 Operational Management Layer

The operational mangement layer is responsible to handle all the arctifacts required by the factory to perform the required processing requirements. It also bounds the factory to only be able to perform functions the layer already manages.

Autonomous Node Transport (ANT) Definitions

The autonomous node transport definitions are the set of cloud instances or physical servers configurations supporting the processing capability of the factory. The HORUS schema keeps a series of characteristics required by the factory to decide which autonomous node transport to use for which requirement. The nodes will be changed in the future as characteristics for new nodes are discovered and loaded into the factory.

The nodes are heterogeneous computing enabled and support combinations of heterogeneous processors that covers the range from high-end servers and high-performance computing machines to low-power embedded devices like mobile phones and tablets.

Autonomous Node Transport Management

The autonomous node transport management oversees the complete process of running the heterogeneous computing systems. The combinations and work flow of the nodes is pre-defined in this section.

Monitoring

Monitoring handles the monitoring tasks in the system. The monitoring covers all aspects of the factory's performance.

Persistent Uniform Protocol Agreement (PUPA) Definitions

The persistent uniform protocol agreement definitions are the collection of the algorithmic skeletons within the system.The PUPAs are programs generate using existing frameworks like OpenCL [15], ArrayFire [9], Spark [8], Titan graph database [16] [10], FastFlow Framework [1]. This is the main area of research for the team. The enhancement and creation of new parallel patterns will improve the capasity of the synaptic assimilator to build new factories and process the data into knowledge.

Persistent Uniform Protocol Agreement (PUPA) Management

The persistent uniform protocol agreement management oversees the complete collection. The process ensures that all uniform protocol agreements are manange to achieve the required end goal of the factory to effectively and efficently process data into knowledge.

Alerting

The alerts are manage from this singular point in the system. The alerting process interacts with the communication process to ensure the appropriate response is generated for the alerts.

Parameters

The parameters are stored in this singular place in the system. Managing the parameters in a single location enhance sthe factory to adapt to changing requirements in a rapid time.

Scheduling

The scheduling handles the schedules from this singular point in the system.

Communication

Communication handles the communication into and from the system from this singular point.

2.1.3 Audit, Balance and Control Layer

Work Cells

The work cells [7] is the fundamental building block of the processing system.

The work cell is executing as an actor model (a mathematical model of concurrent computation) that use "actors" as the universal primitives of concurrent computation: in response to a message it receives, an actor can make local decisions, create more actors, send more messages, and determine how to respond to message received.

Execution Statistics

The execution statistics is the fundamental performance recording system for the factory in the solution.

Remote Yoke

The Yoke or Poka-yoke is fundamental process of "mistake-proofing". The Remote York is the rapid information factory's fundamental monitoring interface between the different work cells.

Rejections and Error Handling

The rejections and error handling in the rapid information factory handles the rejections and error handling within the system.

Balancing and Control

The balance and control mechanisms are the execute from the singular section.

Codes Management

The code management is the single section that holds the standard codes used in the system.

Standard codes includes ISO codes, pre-agreed names and known lists of items for the factory.

Example of the ISO standards used are:

- ISO 3166 - Country Codes of the world
- ISO 4217 - Currency Codes of the world
- ISO 6709 - Representation of latitude, longitude and altitude for geographic point locations.
- ISO 639 - Language codes
- ISO 19134:2006 - Multimodal location-based services for routing and navigation.
- ISO 4030:1983 - Vehicle identification number (VIN)

The use of standard codes enable the effective deep data mining prescribed as output of the factory.

2.1.4 Business Layer

Functional Requirements

A functional requirement defines a function of a system and its components. A function is described as a set of inputs, the behavior, and outputs. [13]. The set of requirements together as a unit describes the factory processing rules.

Non-functional Requirements

A non-functional requirement is a requirement that specifies criteria that can be used to test the operation of a factory, rather than specific behaviors of the process within the factory.The set of requirements together as a unit describes the factory verification rules.

The following are types of non-functional requirements

- Accessibility
- Audit and control
- Availability

  Availability of factory as a factor of its reliability.
- Backup

- Capacity (current and forecast)
- Certification

  Certification of the factory can be achieved for several different ISO standards.
- Compliance

  Compliance is achieved against a minimum set of criteria for the factory.
- Configuration management
- Dependency on other parties
- Deployment
- Documentation
- Disaster Recovery
- Efficiency
- Effectivenes
- Emotional factors
- Environmental protection
- Escrow
- Exploitability
- Extensibility
- Failure management
- Fault tolerance
- Legal and licensing issues
- Patent-infringement-avoidability
- Interoperability
- Maintainability

- Modifiability

- Network topology

- Open source

- Operability

- Performance / Response time

  Short response time for a spesific PUPA to complete.

  High throughput in the factory

  Low utilization of computing resources in the factory.

- Platform compatibility

- Price/Cost

- Privacy

- Portability

- Quality

- Recovery / Recoverability

- Reliability

- Reporting

- Resilience

- Resource constraints

- Response time

  Requirement for timely response from the factory.

- Reusability

- Robustness

- Safety

- Scalability

  The horizontal and vertical scalability is the ratio the factory can expand its resources for processing.

- Security
- Software tools
- Stability
- Standards
- Supportability
- Testability
- Usability by user community
- User Friendliness

2.1.5 *Utility Layer*

The utility layer stores processing structures across the factory for general or common requirements.

Maintenance Utilities

The maintenance untilities are procesing structures that perform work for the factory to maintain

Data Utilities

Spesific Utilities

Autonomous Logical Agreement Transport Executor (ALATE)

The autonomous logical agreement transport executor is a special utility that builds a fundamental metadata view of the data source it is processing and configures a fundamental set of PUPA that will form the fundamental factory for the data source.

Rapid Artifical Intelligence Data Extract Routine (RAIDER)

The rapid artifical intelligence data extract routine is a special utility that builds process PUPA for Retrieve of data via NEST PUPA.

Rapid Execute Artificial Protocol Engine for Routine (REAPER)

The rapid execute artificial protocol engine for routine is a special utility that performs a stand alone execution of any pre-approved work cell.

Sequencetial Converter into Ontology for Uniform Transport (SCOUT)

The sequencetial converter into ontology for uniform transport is a special utility that builds NEST PUPA for a data source.

The SCOUT connects to the external data and discovers the metadata required to connect to the data source.

The output is a NEST script using HORUS scripts to be used by the factory to connect to the spesific data source.

2.2 Functional Layer

2.2.1 Retrieve Superstep

The retrieve superstep uses a series of work cells with an assembly format that is made up out of four components:

- Remote Monitor Yoke
- Input PUPA or Input NEST PUPA
- ANT
- Output PUPA

The remote monitoring yoke connects the work cell to the PRISM that controls the spesific factory to enable the communication and control to the PRISM's remote motering yoke to ensure the process is monitored and that it complies to its assigned task in the factory.

The Input PUPA or Input NEST PUPA describes the processsing rules and formats of the data source use as the input to the process. The factory translates the HORUS scripts to generate the processing logic to input the data.

The ANT is the setup script in HORUS rules that builds a processing engine to process the data form the input PUPA into the processing rules of the output PUPA.

The output PUPA is the processsing rules and formats of the data source use as the output from the process. The factory translates the HORUS scripts to generate the processing logic to output the data.

2.2.2 Assess Superstep

The assess superstep uses a series of work cells with an assembly format that is made up out of four components:

- Remote Monitor Yoke
- Input PUPA
- ANT
- Output PUPA

The remote monitoring yoke connects the work cell to the PRISM that controls the spesific factory to enable the communication and control to the PRISM's remote motering yoke to ensure the process is monitored and that it comples it assigned task in the factory.

The Input PUPA NEST PUPA describes the processsing rules and formats of the data source use as the input to the process. The factory translates the HORUS scripts to generate the processing logic to input the data.

The ANT is the setup script in HORUS rules that builds a processing engine to process the data form the input PUPA into the processing rules of the output PUPA.

The output PUPA is the processsing rules and formats of the data source use as the output from the process. The factory translates the HORUS scripts to generate the processing logic to output the data.

2.2.3 Process Superstep

The process superstep uses a series of work cells with an assembly format that is made up out of four components:

- Remote Monitor Yoke
- Input PUPA
- ANT
- Output PUPA

The remote monitoring yoke connects the work cell to the PRISM that controls the spesific factory to enable the communication and control to the PRISM's remote motering yoke to ensure the process is monitored and that it comples it assigned task in the factory.

The Input PUPA NEST PUPA describes the processsing rules and formats of the data source use as the input to the process. The factory translates the HORUS scripts to generate the processing logic to input the data.

The ANT is the setup script in HORUS rules that builds a processing engine to process the data form the input PUPA into the processing rules of the output PUPA.

The output PUPA is the processsing rules and formats of the data source use as the output from the process. The factory translates the HORUS scripts to generate the processing logic to output the data.

2.2.4 Transform Superstep

The transform superstep uses a series of work cells with an assembly format that is made up out of four components:

- Remote Monitor Yoke
- Input PUPA
- ANT
- Output PUPA

The remote monitoring yoke connects the work cell to the PRISM that controls the spesific factory to enable the communication and control to the PRISM's remote motering yoke to ensure the process is monitored and that it comples it assigned task in the factory.

The Input PUPA NEST PUPA describes the processsing rules and formats of the data source use as the input to the process. The factory translates the HORUS scripts to generate the processing logic to input the data.

The ANT is the setup script in HORUS rules that builds a processing engine to process the data form the input PUPA into the processing rules of the output PUPA.

The output PUPA is the processsing rules and formats of the data source use as the output from the process. The factory translates the HORUS scripts to generate the processing logic to output the data.

2.2.5 Organise Superstep

The organise superstep uses a series of work cells with an assembly format that is made up out of four components:

- Remote Monitor Yoke
- Input PUPA
- ANT
- Output PUPA

The remote monitoring yoke connects the work cell to the PRISM that controls the spesific factory to enable the communication and control to the PRISM's remote motering yoke to ensure the process is monitored and that it comples it assigned task in the factory.

The Input PUPA NEST PUPA describes the processsing rules and formats of the data source use as the input to the process. The factory translates the HORUS scripts to generate the processing logic to input the data.

The ANT is the setup script in HORUS rules that builds a processing engine to process the data form the input PUPA into the processing rules of the output PUPA.

The output PUPA is the processsing rules and formats of the data source use as the output from the process. The factory translates the HORUS scripts to generate the processing logic to output the data.

2.2.6 Report Superstep

The report superstep uses a series of work cells with an assembly format that is made up out of four components:

- Remote Monitor Yoke
- Input PUPA
- ANT
- Output PUPA

The remote monitoring yoke connects the work cell to the PRISM that controls the spesific factory to enable the communication and control to the PRISM's remote motering yoke to ensure the process is monitored and that it comples it assigned task in the factory.

The Input PUPA NEST PUPA describes the processsing rules and formats of the data source use as the input to the process. The factory translates the HORUS scripts to generate the processing logic to input the data.

The ANT is the setup script in HORUS rules that builds a processing engine to process the data form the input PUPA into the processing rules of the output PUPA.

The output PUPA is the processsing rules and formats of the data source use as the output from the process. The factory translates the HORUS scripts to generate the processing logic to output the data.

2.3 Work Cells

The remote work cells is the fundamental processing container of the rapid information factory.

2.3.1 Monitor Work Cell

The monitor work cell consists of a persistent recursive information schema manipulator plus a remote assessment yoke for each processing work cell the spesific BSP flow requires in the rapid information factory

2.3.2 Processing Work Cell

The processing work cell is a combination of a remote assessment yoke, an input persistent uniform protocol agreement, an autonomous node transport and an output persistent uniform protocol agreement. The remote assessment yoke communicates to the remote assessment yoke attached to the monitor work cell. The input persistent uniform protocol agreement holds the instructions to enable the work cell to import the data into the work cell. The output persistent uniform protocol agreement holds the instructions to enable the work cell to export the data from the work cell.The autonomous node transport supplies the processing power to execute the PUPA and the yoke instructions.

2.3.3 Measure Work Cell

The measure work cell consists of an autonomous node transport that supplies the processing power and a measure agreement precision that supplies the tests to determine if the processing was successful.

2.4 Rapid Information Factory Data Sources

2.4.1 Retrieve Data Sources

The retrieve data sources are external data source that requires a spesial type of persistent uniform protocol agreement called a node extractor and schema transformer that supplies the data processing instructions to transform the extraernal data into HORUS compliant data structures. The additional data workspace supplies preloaded data to assist the retrieve superstep to load the data from the external data source to create the retrieve data workspace that is the main storage structure in HORUS any retrieve data loads.

2.4.2 Assess Data Sources

The assess data sources are a read only input from the retrieve data workspace, a reference data workspace that is a read only data source for supplying reference data for the assess procudures.Reference data can iclude lists of codes and description that are valid data or lookup data to enhance the quality of the data by adding extra information to the assess data. The assess data workspace is the main storage structure for HORUS data.

2.4.3 Process Data Sources

The process data sources are a read only input from the assess data workspace, a reference data workspace that is a read only data source for supplying reference data for the process procudures.Reference data can iclude lists of codes and description that are valid data or lookup data to enhance the quality of the data by adding extra information to the process data. The process data workspace is the main storage structure for HORUS data processed into a data vault containing hubs, links and satellites.

2.4.4 Data Vault

The Data Vault architecture offers a unique solution to data integration in the rapid information factory. The Data Vault is a detail oriented, historical tracking and uniquely linked set of normalised tables that support one or more functional areas of the factory that stores perfeactly as data island on top of the data lake structure to process unstructured and semi-structured data into structured data.

Benefits of Data Vault Modeling

- Manage and enforce compliance to Sarbanes-Oxley, HIPPA, and BASIL II in factory.

- Identify business data defects that were not visible before the processing.
- Rapidly reduce business cycle time for implementing enhancements.
- Merge new business units into the organisation rapidly.
- Rapid return-on-investment and delivery of information to new star schemas in data warehouse.
- Consolidate disparate data stores in a homogeneous data lake.
- Effective and Efficient Master Data Management.
- Implement and Deploy service-oriented architecture (SOA) via factory.
- Scale to exabytes of data in the data lake.
- SEI CMM Level 5 compliant (Repeatable, consistent, redundant architecture).
- Full data lineage for all data from the source systems.

2.4.5 Transform Data Sources

The transform data sources are a read only input from the process data workspace, a reference data workspace that is a read only data source for supplying reference data for the transform procudures.Reference data can iclude lists of codes and description that are valid data or lookup data to enhance the quality of the data by adding extra information to the transform data. The transform data workspace is the main storage structure for HORUS data warehouse structure that supports any analytic inquiries.

2.4.6 Organise Data Sources

The organise data sources are read only input from the Tranform data workspace, the organise data workspace to handle any organise data manipulation, the rapid information framework for datamarts, the rapid information framework for analytics and the rapid information framework for cubes that is the main storage structures for the factory.

2.4.7 Report Data Sources

The report data sources are read only input from the rapid information framework for datamarts, the rapid information framework for analytics and the rapid information framework for cubes. The role based access contol security process enforces any role based security access to the data sources. The rapid information framework for visualistion handles the factory's visualisation requirements. The rapid information framework for exports are the export methord for the rapid information factory and formats the HORUS compliant data structures into external data formats via a persistent uniform protocol agreement.

3. RAPID TEST FRAMEWORK

3.1 Unit testing

3.1.1 Static Testing

YOKE Unit Testing

The YOKE unit testing enables the rapid information factory to test all the YOKE structures individually.

3.2 Solution Testing

The solution testing performance the testing of the solution.

Solution Testing Plan

The solution testing plan is the process description of how to test the solution as a complete factory.

3.2.1 Link Testing

Generate Link Test Data

Prepare data for each Link Test to match the spesific measure agreement precision instructions.

Execute Singular Link Test

Execute the Link Test instructions by combining a remote assessment yoke, an appropiate autonomous node transport and the spesific measure agreement precision.

Execute Parallel Link Test

Execute the Link Test instructions in parallel by combining a remote assessment yoke, an appropiate autonomous node transport and the spesific measure agreement precision.

3.2.2 System Testing

Generate System Test Data

Prepare data for each System Test to match the spesific measure agreement precision instructions.

Execute Singular System Test

Execute the System Test instructions by combining a remote assessment yoke, an appropiate autonomous node transport and the spesific measure agreement precision.

Execute Parallel System Test

Execute the System Test instructions in parallel by combining a remote assessment yoke, an appropiate autonomous node transport and the spesific measure agreement precision.

3.2.3 Performance Testing

Generate Performance Test Data

Prepare data for each Performance Test to match the spesific measure agreement precision instructions.

Execute Singular Performance Test

Execute the Link Performance instructions by combining a remote assessment yoke, an appropiate autonomous node transport and the spesific measure agreement precision.

Execute Parallel Performance Test

Execute the Performance Test instructions in parallel by combining a remote assessment yoke, an appropiate autonomous node transport and the spesific measure agreement precision.

Solution Completion Report

The solution completion report is the combined data for the solution testing.

3.3 Acceptance Testing

The acceptance testing performance the testing of the solution.

Acceptance Testing Plan

The acceptance testing plan is the process description of how to test the solution for acceptance by the users.

3.3.1 Acceptance Testing

Generate Acceptance Test Data

Prepare data for each acceptance test to match the spesific measure agreement precision instructions.

Execute Singular Acceptance Test

Execute the acceptance test instructions by combining a remote assessment yoke, an appropiate autonomous node transport and the spesific measure agreement precision.

Execute Parallel Acceptance Test

Execute the acceptance test instructions in parallel by combining a remote assessment yoke, an appropiate autonomous node transport and the spesific measure agreement precision.

3.3.2 Exploratory Parallel Testing

Generate Exploratory Parallel Test Data

Prepare data for each exploratory parallel test to match the spesific measure agreement precision instructions.

Execute Singular Exploratory Parallel Test

Execute the exploratory parallel test instructions by combining a remote assessment yoke, an appropiate autonomous node transport and the spesific measure agreement precision.

Execute Parallel Exploratory Parallel Test

Execute the exploratory parallel test instructions in parallel by combining a remote assessment yoke, an appropiate autonomous node transport and the spesific measure agreement precision.

Acceptance Completion Report

The acceptance completion report is the combined data for the solution testing.

4. PROCESS LAYER - DATA VAULT

A data vault consists of three types of data structures:

- Hub
- Satellite
- Link

4.1 Time-People-Object-Location-Event

4.1.1 Time (Hub)

The time hub contains at the following fields:

- a surrogate key, use for connection by the rest of the data structures in the data vault.
- a business key, the business key combined from date and time for the time hub.
- the record source that can be used for lineage of the date and time keys.
- the metadata fields with information for manual updates (user/time) and the extraction date.

4.1.2 Time Details (Satellite)

The time satellite contains at the following fields:

- a surrogate key from the Time Hub
- a ISO 8601 data format for the date and time

4.1.3 People (Hub)

The people hub contains at the following fields:

- a surrogate key, use for connection by the rest of the data structures in the data vault.
- a business key combined from people's first name, middle name, last name and date-of-birth for the people hub.
- the record source that can be used for lineage of the people keys.
- the metadata fields with information for manual updates (user/time) and the extraction date.

4.1.4 People Details (Satellite)

??????

4.1.5 Object (Hub)

The object hub contains at the following fields:

- a surrogate key, use for connection by the rest of the data structures in the data vault.
- a business key combined from object type and object identifier for the object hub.
- the record source that can be used for lineage of the object keys.
- the metadata fields with information for manual updates (user/time) and the extraction date.

4.1.6 Object Details (Satellite)

????

4.1.7 Location (Hub)

The ISO 6709 Standard representation of geographic point location by coordinates.

The location hub contains at the following fields:

- a surrogate key, use for connection by the rest of the data structures in the data vault.
- a business key combined from latitude, longitude and altitude for geographic point locations for the location hub.
- the record source that can be used for lineage of the location keys.
- the metadata fields with information for manual updates (user/time) and the extraction date.

4.1.8 Location Details (Satellite)

????

4.1.9 Event (Hub)

The event hub contains at the following fields:

- a surrogate key, use for connection by the rest of the data structures in the data vault.
- a business key combined from event type and event identifier for the event hub.
- the record source that can be used for lineage of the event keys.
- the metadata fields with information for manual updates (user/time) and the extraction date.

4.1.10 Event Details (Satellite)

????

4.1.11 Time People (Link)

????

4.1.12 Time Object (Link)

????

4.1.13 Time Location (Link)

????

4.1.14 Time Event (Link)

????

4.1.15 People Object (Link)

????

4.1.16 People Location (Link)

????

4.1.17 People Event (Link)

????

4.1.18 People People (Link)

????

4.1.19 Object Location (Link)

????

4.1.20 Object Event (Link)

????

4.1.21 Object object (Link)

????

4.1.22 Location Event (Link)

????

4.1.23 Location location (Link)

????

4.1.24 Event Event (Link)

????

5. TRANSFORM LAYER - SUN MODELS

The transform layer converts the data into business aligned knowledge structures that the business can formulate queries to answer business questions.

5.1 Dimensions

The dimension is the information component of the business structure.

5.1.1 Type 0 Dimension

The Type 0 method is passive as it only insert but never updates.

5.1.2 Type 1 Dimension

The Type 1 method is active as it overwrites old with new data but keeps no track of historical data.

5.1.3 Type 2 Dimension

The Type 2 method is active as it tracks historical data by creating multiple records for a given natural key and keeps track of period the values was valid.

5.1.4 Outrigger Dimension

The outrigger dimension contain a reference to another dimension table to enable linking the two dimensions attributes.

5.1.5 Bridge Dimension

The bridge dimension is mapping solution that resolves many-to-many relationship within the dimension schema by converting it to a one-to-many and a many-to-one relationship.

5.1.6 Mini Dimension

The mini dimension creates a new structure that moves dimensional attributes that are rapidly changing into a miniâĂŞdimension to split it from non-rapid changing dimensional attributes.

5.2 Facts

A fact table consists of the measurements, metrics for business process.

5.2.1 Measures

Measures are facts that store measurements and metrics that enable the apllication of mathematic process on the values.

5.2.2 Factless

Factless facts are spesial structures that only consist of keys that links to the relavent dimensions to the facts.

6. SCHEDULE FRAMEWORK

6.1 Cycle Time

Cycle time describes the time use to complete a specific task from start to finish.

The time is calculated as:.

$$CycleTime = ProcessSetupTime + ProcessTime + ProcessRecycleTime + VerifySetupTime + VerifyTime + VerifyRecycleTime \quad (1)$$

6.2 Value Stream

Value stream mapping is the lean-management method for analysing the current state and designing a improved state for the series of activities.

6.3 Value Added Procesing Time

The value added processing time is the time the factory process the data to add value.

6.4 Non Value Added Procesing Time

The non value added processing time is the time the factory wastes while process the data.

6.5 Production Lead Time

Production Lead Time is the total time (Value Added Procesing and Non Value Added Procesing) use to execute the data through an entire value stream.

6.6 Schedule Backlog

The Schedule Backlog is all the required processing Persistent Uniform Protocol Agreements (PUPA) prioritised, ordered list, sorted by business value and risk. It contains the tasks to accomplish the RAPTOR flow. The Schedule Backlog often contains user stories covering functional requirements, non-functional requirements for the RAPTOR flow.

6.6.1 INVEST RAPTOR flow

Independent

RAPTOR flow must be independent. The PUPA must ensure that it does not interfere with other PUPAs.

Negotiable

A RAPTOR flow always negotiable. It is not an explicit contract for functionality of the PUPA. The details will be co-created by the customer and programmer during development. The improvement process will ten ensure the output is delivered with maximum efficiency.

Valuable

A RAPTOR flow must be valuable to business. No non-value add PUPA will execute in the factory.

Estimable

A RAPTOR flow must be estimated. Any PUPA must have a estimate for execution.

Small

RAPTOR flow must be as small as possible. Each PUPA should only perform singular functions.

Testable

A RAPTOR flow must be testable. Each PUPA should be testable for a known input to a know output.

6.6.2 SMART Tasks

Specific

A RAPTOR flow task needs to be specific and tasks to add up to the full RAPTOR flow.

Measurable

The key measure is testability for successful end-to-end results.

Achievable

The task should be achievable. Each PUPA must hold a achievable function.

Relevant

Every PUPA/task should be relevant and contributing to the RAPTOR flow in the factory. Stories are divided into tasks for the achievement of RAPTOR flow.

Time-boxed

A PUPA/task should be time-boxed: limited to a specific duration.

6.7 Active Process Backlog

The active process backlog consists of the committed process tasks attached to the scheduled PRISM controlled end-to-end RAPTOR flows. The backlog uses a Drum âĂŞ Buffer âĂŞ Rope (DBR) scheduling methodology [6] that monitors the workcells for progress and then commit the next PUPA into the process.

6.8 Active Process Work Cells

The active process work cells is the combination of processing structures that is currently active in the rapid information factory. The quantity is determined by the available and required parallel processing units active in the factory.

6.8.1 Process Set-up Time

The process set-up time is the time it takes the factory to construct the work cell and bring it online ready for processing data.

6.8.2 Process Run Time

The process run time is the time the work cell uses to process the data against the spesific input and output PUPAs' algoritmes.

6.8.3 Process Reset Time

The process reset time is the time it takes the factory to reset the spesific work cell ready for the next work cell.

6.9 Verify Backlog

The verify backlog consists of the committed process verify tasks attached to the scheduled PRISM controlled end-to-end RAPTOR flows.

6.10 Active Verify Backlog

The active verify backlog consists of the committed verify tasks attached to the scheduled PRISM controlled end-to-end RAPTOR flows.The backlog uses a Drum âĂŞ Buffer âĂŞ Rope (DBR) scheduling methodology [6] that monitors the workcells for progress and then commit the next PUPA into the process.

6.11 Active Verify Work Cells

The active verify work cells is the combination of processing verify structures that is currently active in the rapid information factory. The quantity is determined by the available and required parallel processing units active in the factory.

6.11.1 Verify Set-up Time

The verify set-up time is the time it takes the factory to construct the work cell and bring it online ready for processing verification data.

6.11.2 Verify Run Time

The verify run time is the time the work cell uses to verify the data against the spesific MAP's algoritmes.

6.11.3 Verify Reset Time

The verify reset time is the time it takes the factory to reset the spesific work cell ready for the next work cell.

6.12 Information Process Log

The information process log consists of the completed process and verification tasks attached to the scheduled PRISM controlled end-to-end RAPTOR flows.

7. IMPROVEMENT PROCESSES

7.1 The 8 Wastes

7.1.1 Defects.

Defects are mistakes that require extra time, resources and money to reprocess.

The defects are the result of:

- Poor quality control of processes (ANTs, PUPAs).
- Poor repairs on servers.
- Poor documentation procedures in HORUS.
- Lack of standards in HORUS.
- Weak or missing processes (ANTs, PUPAs).
- A misunderstanding of customer requirements.
- Poor inventory control (ANTs, PUPAs and Data Lake).
- Poor design of processes (ANTs, PUPAs and Data Lake).
- Undocumented design changes (ANTs, PUPAs and Data Lake).

7.1.2 Overproduction.

Overproduction is when too many of a spesific deliverable is delivered.

The factory would cause overproduction if too many ANTs is prepared for running PUPAs:

- Just-in-case production, processing data without needing it.
- Unclear customer requirements, RAPTOR PUPA's process defective data.
- Producing to a incorrect forecast, processing is done at incorrect time.
- Long set-up times, ANTs take to long to start-up ready for processing.
- Attempts to avoid long set-up times, leaving ANT's running just incase they are needed.
- Poorly applied automation, the process does not perform as required.

7.1.3 Waiting

Actual downtime that occurs whenever processing has to stop for an unplanned/planned reason.

Causes of waiting can also include:

- Mismatched production rates, data arrives too early or too late.
- Very long set-up times, ANT's take too long to be ready for processing.
- Poor RAPTOR workflow, design to workflow for optimum throughput.
- Insufficient staffing, when human intervention is need it has to be ready at correct time.
- Work absences, if the data is dependent on human intervention to complete a task then absences will cause waiting.
- Poor communications, if the yokes loses communication the process will cause waiting or lack of synchronisation.

7.1.4 Non-Utilised Talent

Non-Utilised Talent is the poor utilization of available talents, ideas, abilities and skill sets.

- Lack of teamwork between staff members.
- Lack of training in operations of the factory.
- Poor communications between staff members and the factory.
- Management's refusal to include employees in problem-solving.
- Narrowly defined jobs and expectations for staff.
- Poor management of staff.

7.1.5 Transportation

Transportation is the unnecessary moving data around within the factory.

- Poor factory layout via HORUS.
- Excessive or unnecessary handling of data via PUPAs.
- Misaligned process flow in the factory via HORUS.
- Poorly-designed factory PUPAs via HORUS.
- Unnecessary steps in factory PUPA processes via HORUS.

7.1.6 Inventory

Lean are based on the practice of Just-In-Time production of data in the factory.

Excess inventory is caused by:

- Overproduction.
- Poor layout.
- Mismatched production speeds.
- Unreliable suppliers of data.
- Long set-up times of ANTs.
- Misunderstood customer requirements.

7.1.7 Motion

Excess motion is to move around too much and then causes the factory slow down significantly.

Causes of excessive motion include:

- Poor work cell and workflow in the RAPTOR flow.
- Poor housekeeping of ANT's recycle process.
- Shared ANT and PUPA processing.
- Work cell congestion
- Isolated operations in PUPA.
- Lack of standards in PUPA.
- Poor process design and controls of PUPA.

7.1.8 Extra-processing

Excess Processing is any unnecessary effort expended in order to complete a task: double-handling data, seeking permission during processing, unnecessary processing steps, unnecessary data useage, re-entering data, making too many copies of data.

Excess Processing arises from:

- Poor process control of the RAPTOR flow via HORUS.
- Lack of standards in PUPA.
- Poor communication between yoke and PRISM.
- Overdesigned equipment for ANT requirements.
- Misunderstanding of the customer's requirements via HORUS.
- Human error.
- Producing to forecast not requirement by factory via HORUS.

7.2 Plan-Do-Act-Check Improvement Process

7.2.1 Plan

Establish the objectives and processes necessary to deliver results in accordance with the expected output (the target or goals). By establishing output expectations, the completeness and accuracy of the specifications is also a part of the targeted improvement. When possible start on a small scale to test possible effects.

7.2.2 Do

Implement the plan, execute the process, execute the factory. Collect data for charting and analysis in the following "CHECK" and "ACT" steps.

7.2.3 Check

Study the actual results (measured and collected in "DO" above) and compare against the expected results (targets or goals from the "PLAN") to ascertain any differences. Look for deviation in implementation from the plan and also look for the appropriateness and completeness of the plan to enable the execution, i.e., "Do". Charting data can make this much easier to see trends over several PDCA cycles and in order to convert the collected data into information. Information is what you need for the next step "ACT".

7.2.4 Act

If the CHECK shows that the PLAN that was implemented in DO is an improvement to the prior standard (baseline), then that becomes the new standard (baseline) for how the factory should ACT going forward (new standards are enACTed). If the CHECK shows that the PLAN that was implemented in DO is not an improvement, then the existing standard (baseline) will remain in place. In either case, if the CHECK showed something different than expected (whether better or worse), then there is more learning to be done and that will suggest potential future PDCA cycles. Note that some who teach PDCA assert that the ACT involves making adjustments or corrective actions. It is a repeating process that improves the factory.

7.3 Define-Measure-Analyse-Improve-Control Improvement Process

7.3.1 Define

The purpose of this step is to clearly articulate the business problem, goal, potential resources, project scope and high-level project timeline. This information is typically captured within project charter document.

7.3.2 Measure

The purpose of this step is to objectively establish current baselines as the basis for improvement. This is a data collection step, the purpose of which is to establish process performance baselines.

7.3.3 Analyse

The purpose of this step is to identify, validate and select root cause for elimination.

7.3.4 Improve

The purpose of this step is to identify, test and implement a solution to the problem; in part or in whole.

7.3.5 Control

The purpose of this step is to sustain the gains. Monitor the improvements to ensure continued and sustainable success. Create a control plan.

7.4 Lean Six Sigma: 5S

7.4.1 Sort

The first step is to go through all equipment (ANTs and PUPAs) and data in the factory and determine what must be retained in the factory. Only essential PRISMs, ANTs and PUPA are allowed to remain. Archive any unwanted entities.

7.4.2 Set in Order

âĂIJA place for everything, and everything in its place.âĂİ

Work through the HORUS structures and ensure all the PRISMs, ANTs and PUPAs are in correct workflows. Use a directed acyclic graph (DAG) to ensure the workflow has a logical start and end. Simulate the process to run through without data to verify the factory before processing happens.

7.4.3 Shine

Monitor and maintain the processes the synaptic assimilator setup in HORUS, thoroughly clean everything remaining in the factories.Ensure effective and efficient execution of PRISMs, ANTs and PUPAs in the factories.

7.4.4 Standardise

Make factories consistent. All PRISMs and PUPAs should be identical so that any ANTs can immediately get initialise and productively execute the process if necessary.

7.4.5 Systematise

This final step means to attach a schedule to use the 5S-ed factory flows.

7.5 Rapid Information Factory Cluster (RIFC)

7.5.1 3D Torus Network Framework

Torus networks are use by top-performing supercompute because it ensures that node in the cluster can directly communicate with the other nodes. A 3D Torus Network enbles a minimum hop network.. The network ha sthe following advantages:

- Ultra low latency message exchange
- Extremely high hardware message rate
- Low memory footprint
- Switchless Design - 3D Torus Direct Network
- High scalability
- Single chip solution
- Extremely efficient, pipelined hardware architecture

7.5.2 MapR Data Lake

The rapid information factory cluster is a bulk synchronous parallel (BSP) engine. The cluster consisting of two hunderd thousand amazon cloud nodes (d2.8xlarge with thirty six processing cores, two hunderd forty four gigabyte memory and twenty four two thousand gigabyte hard disks) to support quintillion calculations per second against quintillion of disk storage.

The hunderd amazon graphical enhanced nodes (g2.8xlarge with four GPUs each with one thousand five hunderd CUDA cores, four gigabyte of video memory, thirty two cpus, sixty gigabyte memory and two hunderd and forty gigabyte solid state drives.)

7.5.3 Titan Graph Data Lake

The two hunderd amazon graphical enhanced nodes (g2.8xlarge with four GPUs each with one thousand five hunderd CUDA cores, four gigabyte of video memory, thirty two cpus, sixty gigabyte memory and two hunderd and forty gigabyte solid state drives.)

7.5.4 Cassandra Data Lake

The five hunderd amazon graphical enhanced nodes (g2.8xlarge with four GPUs each with one thousand five hunderd CUDA cores, four gigabyte of video memory, thirty two cpus, sixty gigabyte memory and two hunderd and forty gigabyte solid state drives.)

8. EXPERIMENTS

8.1 Autonomous Node Transport - Build Test

8.1.1 Amazon Web Services ANTs

A001000001 - t2.micro - Amazon Cloud instance with 1 CPUs and 1 Memory (GB)

A001000002 - t2.small - Amazon Cloud instance with 1 CPUs and 2 Memory (GB)

A001000003 - t2.medium - Amazon Cloud instance with 2 CPUs and 4 Memory (GB)

A001000004 - t2.large - Amazon Cloud instance with 2 CPUs and 8 Memory (GB)

A001000005 - m4.large - Amazon Cloud instance with 2 CPUs and EBS-only Memory (GB) with 450 mbps Throughput

A001000006 - m4.xlarge - Amazon Cloud instance with 4 CPUs and EBS-only Memory (GB) with 750 mbps Throughput

A001000007 - m4.2xlarge - Amazon Cloud instance with 8 CPUs and EBS-only Memory (GB) with 1000 mbps Throughput

A001000008 - m4.4xlarge - Amazon Cloud instance with 16 CPUs and EBS-only Memory (GB) with 2000 mbps Throughput

A001000009 - m4.10xlarge - Amazon Cloud instance with 40 CPUs and EBS-only Memory (GB) with 4000 mbps Throughput

A001000010 - m3.medium - Amazon Cloud instance with 1 CPUs and 3.75 Memory (GB) with 1 x 4 GB SSD

A001000011 - m3.large - Amazon Cloud instance with 2 CPUs and 7.5 Memory (GB) with 1 x 32 GB SSD

A001000012 - m3.xlarge - Amazon Cloud instance with 4 CPUs and 15 Memory (GB) with 2 x 40 GB SSD

A001000013 - m3.2xlarge - Amazon Cloud instance with 8 CPUs and 30 Memory (GB) with 2 x 80 GB SSD

A001000014 - c4.large - Amazon Cloud instance with 2 CPUs and 3.75 Memory (GB) with 500 mbps Throughput

A001000015 - c4.xlarge - Amazon Cloud instance with 4 CPUs and 7.5 Memory (GB) with 750 mbps Throughput

A001000016 - c4.2xlarge - Amazon Cloud instance with 8 CPUs and 15 Memory (GB) with 1000 mbps Throughput

A001000017 - c4.4xlarge - Amazon Cloud instance with 16 CPUs and 30 Memory (GB) with 2000 mbps Throughput

A001000018 - c4.8xlarge - Amazon Cloud instance with 36 CPUs and 60 Memory (GB) with 4000 mbps Throughput

A001000019 - c3.large - Amazon Cloud instance with 2 CPUs and 3.75 Memory (GB) with 2 x 16 GB SSD

A001000020 - c3.xlarge - Amazon Cloud instance with 4 CPUs and 7.5 Memory (GB) with 2 x 40 GB SSD

A001000021 - c3.2xlarge - Amazon Cloud instance with 8 CPUs and 15 Memory (GB) with 2 x 80 GB SSD

A001000022 - c3.4xlarge - Amazon Cloud instance with 16 CPUs and 30 Memory (GB) with 2 x 160 GB SSD

A001000023 - c3.8xlarge - Amazon Cloud instance with 32 CPUs and 60 Memory (GB) with 2 x 320 GB SSD

A001000024 - r3.large - Amazon Cloud instance with 2 CPUs and 15.25 Memory (GB) with 1 x 32 GB SSD

A001000025 - r3.xlarge - Amazon Cloud instance with 4 CPUs and 30.5 Memory (GB) with 1 x 80 GB SSD

A001000026 - r3.2xlarge - Amazon Cloud instance with 8 CPUs and 61 Memory (GB) with 1 x 160 GB SSD

A001000027 - r3.4xlarge - Amazon Cloud instance with 16 CPUs and 122 Memory (GB) with 1 x 320 GB SSD

A001000028 - r3.8xlarge - Amazon Cloud instance with 32 CPUs and 244 Memory (GB) with 2 x 320 GB SSD

A001000029 - g2.2xlarge - Amazon Cloud instance with 1 GPUs High-performance NVIDIA GPUs, each with 1,536 CUDA cores and 4GB of video memory, 8 CPUs and 15 Memory (GB) with 1 x 60 GB SSD

A001000030 - g2.8xlarge - Amazon Cloud instance with 4 GPUs High-performance NVIDIA GPUs, each with 1,536 CUDA cores and 4GB of video memory, 32 CPUs and 60 Memory (GB) with 2 x 120 GB SSD

A001000031 - i2.xlarge - Amazon Cloud instance with 4 CPUs and 30.5 Memory (GB) with 1 x 800 SSD GB SSD

A001000032 - i2.2xlarge - Amazon Cloud instance with 8 CPUs and 61 Memory (GB) with 2 x 800 SSD GB SSD

A001000033 - i2.4xlarge - Amazon Cloud instance with 16 CPUs and 122 Memory (GB) with 4 x 800 SSD GB SSD

A001000034 - i2.8xlarge - Amazon Cloud instance with 32 CPUs and 244 Memory (GB) with 8 x 800 SSD GB SSD

A001000035 - d2.xlarge - Amazon Cloud instance with 4 CPUs and 30.5 Memory (GB) with 3 x 2000 HDD GB SSD

A001000036 - d2.2xlarge - Amazon Cloud instance with 8 CPUs and 61 Memory (GB) with 6 x 2000 HDD GB SSD

A001000037 - d2.4xlarge - Amazon Cloud instance with 16 CPUs and 122 Memory (GB) with 12 x 2000 HDD GB SSD

A001000038 - d2.8xlarge - Amazon Cloud instance with 36 CPUs and 244 Memory (GB) with 24 x 2000 HDD GB SSD

8.1.2 Google Cloud Platform ANTs

ANT002000001 - App Engine

ANT002000002 - Compute Engine

ANT002000003 - Container Engine

ANT002000004 - Cloud Storage

ANT002000005 - Cloud Datastore

ANT002000006 - Cloud SQL

ANT002000007 - Cloud Big Table

ANT002000008 - BigQuery

ANT002000009 - Cloud Dataflow

ANT002000010 - Cloud Pub/Sub

ANT002000010 - Cloud Endpoints

ANT002000010 - Translate API

ANT002000010 - Prediction API

8.1.3 OpenStack ANTs

ANT003000001 - ?????

Redhat 2 Intel Haswell-EP processors, 24 cores, 64GB memory @2133 MHz, and seven available PCI gen3 slots

ANT003000002 - ?????

Ubuntu OpenStack. See Openstack

ANT003000003 - ?????

ANT003000004 - ?????

8.1.4 Physical Cluster ANTs

ANT004000001 - ?????

ANT004000002 - ?????

ANT004000003 - ?????

ANT004000004 - ?????

8.2 Persistent Uniform Protocol Agreements - Build Test

8.2.1 Amazon Web Services ANTs

A001000001 - t2.micro - Amazon Cloud instance with 1 CPUs and 1 Memory (GB)

A001000002 - t2.small - Amazon Cloud instance with 1 CPUs and 2 Memory (GB)

A001000003 - t2.medium - Amazon Cloud instance with 2 CPUs and 4 Memory (GB)

A001000004 - t2.large - Amazon Cloud instance with 2 CPUs and 8 Memory (GB)

A001000005 - m4.large - Amazon Cloud instance with 2 CPUs and EBS-only Memory (GB) with 450 mbps Throughput

A001000006 - m4.xlarge - Amazon Cloud instance with 4 CPUs and EBS-only Memory (GB) with 750 mbps Throughput

A001000007 - m4.2xlarge - Amazon Cloud instance with 8 CPUs and EBS-only Memory (GB) with 1000 mbps Throughput

A001000008 - m4.4xlarge - Amazon Cloud instance with 16 CPUs and EBS-only Memory (GB) with 2000 mbps Throughput

A001000009 - m4.10xlarge - Amazon Cloud instance with 40 CPUs and EBS-only Memory (GB) with 4000 mbps Throughput

A001000010 - m3.medium - Amazon Cloud instance with 1 CPUs and 3.75 Memory (GB) with 1 x 4 GB SSD

A001000011 - m3.large - Amazon Cloud instance with 2 CPUs and 7.5 Memory (GB) with 1 x 32 GB SSD

A001000012 - m3.xlarge - Amazon Cloud instance with 4 CPUs and 15 Memory (GB) with 2 x 40 GB SSD

A001000013 - m3.2xlarge - Amazon Cloud instance with 8 CPUs and 30 Memory (GB) with 2 x 80 GB SSD

A001000014 - c4.large - Amazon Cloud instance with 2 CPUs and 3.75 Memory (GB) with 500 mbps Throughput

A001000015 - c4.xlarge - Amazon Cloud instance with 4 CPUs and 7.5 Memory (GB) with 750 mbps Throughput

A001000016 - c4.2xlarge - Amazon Cloud instance with 8 CPUs and 15 Memory (GB) with 1000 mbps Throughput

A001000017 - c4.4xlarge - Amazon Cloud instance with 16 CPUs and 30 Memory (GB) with 2000 mbps Throughput

A001000018 - c4.8xlarge - Amazon Cloud instance with 36 CPUs and 60 Memory (GB) with 4000 mbps Throughput

A001000019 - c3.large - Amazon Cloud instance with 2 CPUs and 3.75 Memory (GB) with 2 x 16 GB SSD

A001000020 - c3.xlarge - Amazon Cloud instance with 4 CPUs and 7.5 Memory (GB) with 2 x 40 GB SSD

A001000021 - c3.2xlarge - Amazon Cloud instance with 8 CPUs and 15 Memory (GB) with 2 x 80 GB SSD

A001000022 - c3.4xlarge - Amazon Cloud instance with 16 CPUs and 30 Memory (GB) with 2 x 160 GB SSD

A001000023 - c3.8xlarge - Amazon Cloud instance with 32 CPUs and 60 Memory (GB) with 2 x 320 GB SSD

A001000024 - r3.large - Amazon Cloud instance with 2 CPUs and 15.25 Memory (GB) with 1 x 32 GB SSD

A001000025 - r3.xlarge - Amazon Cloud instance with 4 CPUs and 30.5 Memory (GB) with 1 x 80 GB SSD

A001000026 - r3.2xlarge - Amazon Cloud instance with 8 CPUs and 61 Memory (GB) with 1 x 160 GB SSD

A001000027 - r3.4xlarge - Amazon Cloud instance with 16 CPUs and 122 Memory (GB) with 1 x 320 GB SSD

A001000028 - r3.8xlarge - Amazon Cloud instance with 32 CPUs and 244 Memory (GB) with 2 x 320 GB SSD

A001000029 - g2.2xlarge - Amazon Cloud instance with 1 GPUs High-performance NVIDIA GPUs, each with 1,536 CUDA cores and 4GB of video memory, 8 CPUs and 15 Memory (GB) with 1 x 60 GB SSD

A001000030 - g2.8xlarge - Amazon Cloud instance with 4 GPUs High-performance NVIDIA GPUs, each with 1,536 CUDA cores and 4GB of video memory, 32 CPUs and 60 Memory (GB) with 2 x 120 GB SSD

A001000031 - i2.xlarge - Amazon Cloud instance with 4 CPUs and 30.5 Memory (GB) with 1 x 800 SSD GB SSD

A001000032 - i2.2xlarge - Amazon Cloud instance with 8 CPUs and 61 Memory (GB) with 2 x 800 SSD GB SSD

A001000033 - i2.4xlarge - Amazon Cloud instance with 16 CPUs and 122 Memory (GB) with 4 x 800 SSD GB SSD

A001000034 - i2.8xlarge - Amazon Cloud instance with 32 CPUs and 244 Memory (GB) with 8 x 800 SSD GB SSD

A001000035 - d2.xlarge - Amazon Cloud instance with 4 CPUs and 30.5 Memory (GB) with 3 x 2000 HDD GB SSD

A001000036 - d2.2xlarge - Amazon Cloud instance with 8 CPUs and 61 Memory (GB) with 6 x 2000 HDD GB SSD

A001000037 - d2.4xlarge - Amazon Cloud instance with 16 CPUs and 122 Memory (GB) with 12 x 2000 HDD GB SSD

A001000038 - d2.8xlarge - Amazon Cloud instance with 36 CPUs and 244 Memory (GB) with 24 x 2000 HDD GB SSD

8.2.2 Google Cloud Platform ANTs

ANT002000001 - App Engine

ANT002000002 - Compute Engine

ANT002000003 - Container Engine

ANT002000004 - Cloud Storage

ANT002000005 - Cloud Datastore

ANT002000006 - Cloud SQL

ANT002000007 - Cloud Big Table

ANT002000008 - BigQuery

ANT002000009 - Cloud Dataflow

ANT002000010 - Cloud Pub/Sub

ANT002000010 - Cloud Endpoints

ANT002000010 - Translate API

ANT002000010 - Prediction API

8.2.3 OpenStack ANTs

ANT003000001 - ?????

ANT003000002 - ?????

ANT003000003 - ?????

ANT003000004 - ?????

8.2.4 Physical Cluster ANTs

ANT004000001 - ?????

ANT004000002 - ?????

ANT004000003 - ?????

ANT004000004 - ?????

8.3 Persistent Uniform Protocol Agreement - Process (Fundamental)

8.3.1 PUPA001000001 - Sort 1,000 keys assending

8.3.2 PUPA001000002 - Sort 1,000,000 keys assending

8.3.3 PUPA001000003 - Sort 1,000,000,000 keys assending

8.3.4 PUPA001000004 - Sort 1,000 keys across 10 nodes assending

8.3.5 PUPA001000005 - Sort 1,000,000 keys across 10 nodes assending

8.3.6 PUPA001000006 - Sort 1,000,000,000 keys 10 nodes assending

8.4 Persistent Uniform Protocol Agreement - Process (Advance)

The factory should be able to handle the 13 Dwalves of High-Performance Processing [2]

8.4.1 PUPA002000001 - Dense Linear Algebra

The data structure is classic vector and matrix operations that are divided into Level 1 (vector/vector), Level 2 (matrix/vector), and Level 3 (matrix/matrix) operations. Data is normally a contiguous array and computations on elements, rows, columns, or matrix blocks to produce a output.

General dense (all entries nonzero)

Banded (zero below/above some diagonal)

Symmetric/Hermitian

8.4.2 PUPA002000002 - Sparse Linear Algebra

Sparse matrix algorithms are use when input matrices contain a large number of zero entries. Gather-scatter is a type of memory addressing that often arises when addressing vectors in sparse linear algebra operations.

8.4.3 PUPA002000003 - Spectral Methods

Spectral methods are a class of techniques applying mathematics and scientific computing using Fast Fourier Transform.

Data is operated in the spectral domain, often transformed from either a temporal or spatial domain.

8.4.4 PUPA002000004 - N-Body Methods

A n-body problem is the problem of predicting the individual motions of a group of celestial objects interacting with each other gravitationally.

Data is arranged in a regular multidimensional grid (most commonly 2D or 3D).

8.4.5 PUPA002000005 - Structured Grids

A structured Grid is n-dimensional Euclidean space by congruent cells. Example: a 3-dimensional grid is of format m(x,y,z).

8.4.6 PUPA002000006 - Unstructured Grids

A unstructured grid is a grid consisting of vertices and edges that describes a system of interconnected entities.

An irregular mesh or grid, with each grid element being updated from neighboring grid elements.

8.4.7 PUPA002000007 - Monte Carlo Simulations

A problem solving technique used to approximate the probability of outcomes by executing multiple trials, called simulations, using randomised variables.

8.4.8 PUPA002000008 - Combinational Logic

A type of digital logic used by Boolean circuits, where the output is a pure function of the supplied input only.

Combinational logic computations are performing simple operations on very large amounts of data.

8.4.9 PUPA002000009 - Graph Traversal

Graph traversal is the problem of visiting all the nodes in a graph in a particular order.

Graph Traversal applications traverse a number of objects and examine characteristics of those objects to match spesific patterns.

8.4.10 PUPA002000010 - Dynamic Programming

Dynamic programming is a extremely powerful algorithmic paradigm solving a problem by identifying a collection of subproblems and solving them one by one, smallest first, using the results to small problems to solve larger ones, until the whole set of them is solved successfully.

8.4.11 PUPA002000011 - Backtrack and Branch-and-Bound

Branch and bound algorithms are effective for solving various search and global optimization problems. The goal in such problems is to search a very large space to find a globally optimal solution.

8.4.12 PUPA002000012 - Graphical Models

A graphical model or probabilistic graphical model (PGM) is a probabilistic model for a graph expressing the conditional dependence structure between random variables.

Graphical models include Bayesian networks (also known as belief networks, probabilistic networks, causal network, and knowledge maps). Hidden Markov models and neural networks are also graphical models.

8.4.13 PUPA002000013 - Finite State Machine

A finite-state machine (FSM) is a mathematical model of computation used to design computer programs and sequential logic circuits. It is modelled as an abstract machine that can be in one of a finite number of states.

8.5 Persistent Uniform Protocol Agreement - Verify

8.5.1 ?????????????????????????

??????????

8.5.2 ?????????????????????????

??????????

8.5.3 ?????????????????????????

??????????

8.6 Apache Hama

Apache Hama is a distributed computing framework based on Bulk Synchronous Parallel computing techniques for massive scientific computations e.g., matrix, graph and network algorithms.[14]

8.7 Apache Giraph

Giraph is an iterative graph processing system built for high scalability.[5]

8.8 Weaver

A fast and scalable graph store designed specifically for dynamically-changing graphs.[18]

8.9 Medusa

A framework for graph processing using Graphics Processing Units (GPUs) on both shared memory and distributed clusters.[19]

8.10 Titan

Titan is a scalable graph database optimized for storing and querying graphs containing hundreds of billions of vertices and edges distributed across a multi-machine cluster. Titan is a transactional database that can support thousands of concurrent users executing complex graph traversals in real time. It supports ACID and eventual consistency and various big data storage back-ends.[4]

9. RESULTS

?????????????????????????

?????????????????????????

?????????????????????????

????????????????????????

10. REFERENCES

[1] Aldinucci, M., Danelutto, M., Kilpatrick, P., Meneghin, M., and Torquati, M. Accelerating code on multi-cores with fastflow. In *Euro-Par 2011 Parallel Processing*. Springer, 2011, pp. 170–181.

[2] Asanovic, K., Bodik, R., Catanzaro, B. C., Gebis, J. J., Husbands, P., Keutzer, K., Patterson, D. A., Plishker, W. L., Shalf, J., Williams, S. W., et al. The landscape of parallel computing research: A view from berkeley. Tech. rep., Technical Report UCB/EECS-2006-183, EECS Department, University of California, Berkeley, 2006.

[3] Bergman, K., Borkar, S., Campbell, D., Carlson, W., Dally, W., Denneau, M., Franzon, P., Harrod, W., Hill, K., Hiller, J., et al. Exascale computing study: Technology challenges in achieving exascale systems. *Defense Advanced Research Projects Agency Information Processing Techniques Office (DARPA IPTO), Tech. Rep 15* (2008).

[4] Chang, C., Moon, B., Acharya, A., Shock, C., Sussman, A., and Saltz, J. Titan: a high-performance remote-sensing database. In *Data Engineering, 1997. Proceedings. 13th International Conference on* (1997), IEEE, pp. 375–384.

[5] Ching, A. Scaling apache giraph to a trillion edges. *Facebook Engineering blog* (2013).

[6] Daniel, V., and Guide, R. Scheduling with priority dispatching rules and drum-buffer-rope in a recoverable manufacturing system. *International Journal of Production Economics 53*, 1 (1997), 101–116.

[7] Feld, W. M. *Lean manufacturing: tools, techniques, and how to use them.* CRC Press, 2000.

[8] Hintjens, P. Ømq-the guide. *Online: http://zguide. zeromq. org/page: all, Accessed on 23* (2011).

[9] Malcolm, J., Yalamanchili, P., McClanahan, C., Venugopalakrishnan, V., Patel, K., and Melonakos, J. Arrayfire: a gpu acceleration platform. In *SPIE Defense, Security, and Sensing* (2012), International Society for Optics and Photonics, pp. 84030A–84030A.

[10] Mishra, V. Titan graph databases with cassandra. In *Beginning Apache Cassandra Development.* Springer, 2014, pp. 123–151.

[11] O'Leary, D. E. Artificial intelligence and big data. *IEEE Intelligent Systems*, 2 (2013), 96–99.

[12] Ortega, F., Moguerza, J. M., and Cano, E. L. Combining r and python for scientific computing. In *The R User Conference, useR! 2013 July 10-12 2013 University of Castilla-La Mancha, Albacete, Spain* (2013), vol. 10, p. 92.

[13] Roman, G.-C. A taxonomy of current issues in requirements engineering. *Computer 18*, 4 (1985), 14–23.

[14] Seo, S., Yoon, E. J., Kim, J., Jin, S., Kim, J.-S., and Maeng, S. Hama: An efficient matrix computation with the mapreduce framework. In *Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference on* (2010), IEEE, pp. 721–726.

[15] Stone, J. E., Gohara, D., and Shi, G. Opencl: A parallel programming standard for heterogeneous computing systems. *Computing in science & engineering 12*, 1-3 (2010), 66–73.

[16] Tanase, I., Xia, Y., Nai, L., Liu, Y., Tan, W., Crawford, J., and Lin, C.-Y. A highly efficient runtime and graph library for large scale graph analytics. In *Proceedings of Workshop on GRAph Data management Experiences and Systems* (2014), ACM, pp. 1–6.

[17] Team, R. C. R language definition, 2000.

[18] Wu, H., Diamos, G., Cadambi, S., and Yalamanchili, S. Kernel weaver: Automatically fusing database primitives for efficient gpu computation. In *Proceedings of the 2012 45th Annual IEEE/ACM International Symposium on Microarchitecture* (2012), IEEE Computer Society, pp. 107–118.

[19] Zhong, J., and He, B. Medusa: Simplified graph processing on gpus. *Parallel and Distributed Systems, IEEE Transactions on 25*, 6 (2014), 1543–1552.