Hewlett Packard Enterprise

Cray User Group 2025

Search and Query Framework for Workflows with HPC and AI Models

Chris Rickett, Rangan Sukumar and Karlon West

May 08, 2025

Vision: Preparing for the future of AI

Algorithm + Infrastructure

AI success will translate to millions of real-time queries on thousands of AI models

Scale: # of queries

ChatGPT has >400M users
>1 billion queries/day

Query Language + AI agents

The future of data is unstructured, and the future query will be artificially intelligent

Scale: Data and Model size

GPT-3 was trained on ~45 TBs of data and the model size is ~800 GB

Nature + Nurture

If the nature of AI is neural networks, organizations will need a plan for AI nurture

Scale: # of models/query

>500,000 AI models available in Hugging Face (was ~80K in 2022)

Vision: Imagining an enterprise data intelligence platform for the AI era

Insight Interfaces
1000s of users

Plethora of AI models
100s of models

ML-friendly Representation
100s of GBs

Multi-modal Data
10s of TBs

Interactive Dashboards | **Text/Voice Q/A** | **Query/Search engines**

Speech	Vision	Language	Predictive	Generative	What-if ?
WaveNet	ResNet	BERT	SVM	Text-to-Text	Agent-driven
USM	Yolo	Llama-2	Random forests	Text-to-image	Reinforcement
Deepgram	Detectron	Seamless	Regression	Text-to-video	….
…	…	…	…	…	…

Vector embeddings | *Feature store* | *Knowledge graph*

Documents | **Images** | **Time series** | **Video**

Problem: Customers need an “AI-native” data platform

- Business:
  - To experiment and unleash AI innovations for insights on their private/enterprise data like they can with SQL on tabular data today.
  - For natural-language, semantic, approximate and probabilistic search over exact retrieval-based methods using predictive and generative AI.

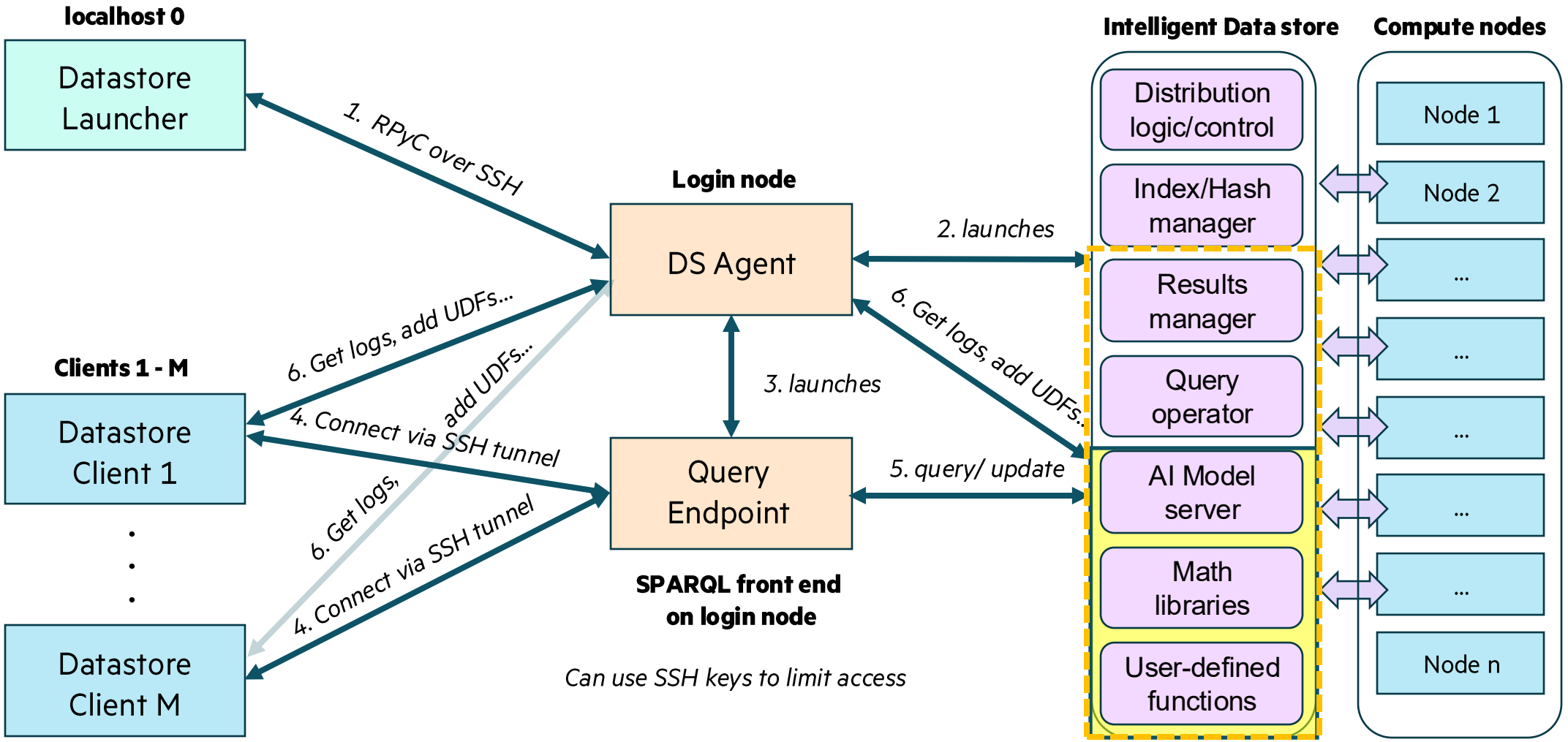
- Technical:
  - Challenges towards building AI-native query engines:
    - Hosting and organizing unstructured data alongside AI models.
    - Load-balancing concurrent queries with multiple AI models.
    - Search/retrieval using set-theoretic and linear algebraic methods.
    - Enabling ingest and query-time bindings/runtimes for AI models.
    - Delivering low latency for both data retrieval and AI inference.

Solution: HPE's Intelligent Data Platform

- Augmenting 10+ years of prior work on the Cray Graph Engine, we have built an intelligent data platform that:
  - Acts as a 3-in-1 multi-modal data, feature and vector store.
  - Unleashes parallel AI inference for faster insights.
  - Speeds-up AI query performance with GPUs.
  - Containerized to run on laptops, VMs and parallel clusters on CSPs, and HPE Supercomputers.
    - Slingshot required libraries built into container for easier deployment

- Ease of Use
  - Python Client API for Developer Interaction
  - Load new data with user programs and AI
  - Import new models and define custom workflows with User-Defined Functions (UDFs)
  - Easily install and manage Python packages to extend capabilities

IDS Architecture



Enabling AI Powered Queries

- Python Client API
  - Enable launch, query, ingest data, import AI models and UDFs, shutdown, etc
  - Interact with a Jupyter notebook or script

- Python Code Import
  - Allow users to import existing python code into running IDS

- User Load Function
  - Enable users to define ‘load’ functions/programs to ingest data in parallel
    - Extract searchable feature set (meta-data) and use to generate graph edges

- Python User Defined Functions
  - Python environment embedded in IDS ranks (each ranks creates at startup)
  - Dynamically load (and reload) python functions
    - Allows users to more easily leverage open-source AI functions

Query Planning for HPC and AI

- Each rank collects profiling information for UDFs, including:
  - Number of times UDF executed
  - Total UDF execution time
  - Number of times UDF resulted in expression rejection
- IDS automatically rebalances solutions across ranks based upon UDF profiling
  - Each rank estimates solutions per second, create ratio compared to slowest and rebalance based on ratio. For example:
    - 1.4 million solutions, 900 ranks:
      - 500 ranks compute 100/second; 300 ranks compute 200/second; 100 ranks compute 300/second
      - 500 + (300 * 2) + (100 * 3) = 1400
      - 1.4 million / 1400 = 10K chunks ➔ 500 slowest each get 10K, 300 ranks get 20K and 100 fastest get 30K
      - 10K / 100 = 100 seconds vs ~14K / 100 = 140 seconds
- Reordering expression evaluations
  - Reorder expressions in logical chains such that fastest UDFs are executed first
- Ability to dynamically optimize query based on AI UDF costs critical
  - User cannot track performance of all models
  - AI inference costs can vary over time based upon input data

How does IDS work as a query engine?

Search query

```
PREFIX arq:
<http://jena.hpl.hp.com/ARQ/function#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-
syntax-ns#>
PREFIX core: <http://purl.uniprot.org/core/>
PREFIX up:<http://purl.uniprot.org/core/>

select ?protB ?name ?mnem ?sim
where {
 # Look up the info for our sars2 spike protein
 ?protein a core:Protein .
 ?protein core:mnemonic 'SPIKE_SARS2' .
 ?protein core:sequence ?isoform .
 ?isoform rdf:value ?seq .


 # Look up all other proteins with sequence info
 ?protB a core:Protein .
 ?protB core:sequence ?isoformB .
 ?isoformB rdf:value ?seqB .
 ?protB core:mnemonic ?mnem .
 ?protB up:recommendedName ?recommended .
 ?recommended up:fullName ?name .


 # Compare the sars2 spike to each protein to get a
sim value
 bind(arq:user_func('ssw', ?seq, ?seqB) as
       ?sim)
 filter(?sim > 0.2)
}
# List the proteins with the highest sim score first
order by desc(?sim)
```

Communication and control between front-end and back-end
SPARQL converter
IP interface to Web Browser
Display or Forward SPARQL and Command results
Generate low-level query (RPN)
Pass Non-SPARQL commands

Login/service/tunnel nodes

Compute nodes

Query Engine Application Images

Image 0 | Operators | Database

Receive, validate and send RPN to all images

SCAN
JOIN
MERGE
OPTIONAL
UNION
FILTER
BIND
...

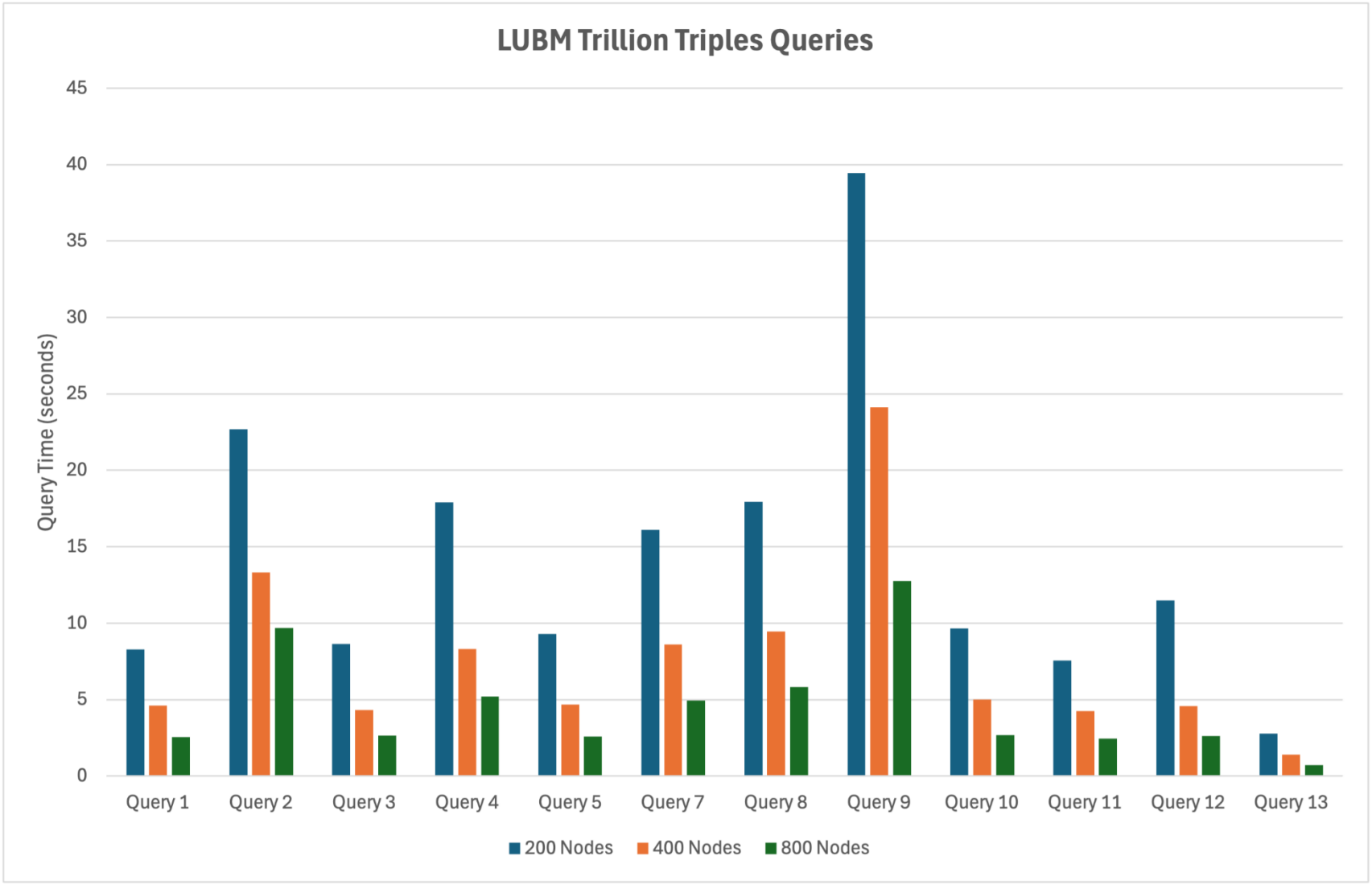
Send results and pointer to output file

Machine Learning models

Implemented as a domain-specific capability into the database

Trillion Triples Knowledge Graph

- Lehigh University Benchmark (LUBM)
  - 14 benchmark queries
  - Query 9 most complex – triangular pattern search across entire graph
- 5.5 million universities
- >1 Trillion edges in the graph
- 128 TB of raw data on disk
- Query 9 and total query times ~10x faster than closest competitor



DEMO #1: Intelligence to tag/annotate/query using pre-trained AI models

```
# Execute a user command to leverage our pretrained ML/DL models to predict objects in a given set of input images
# and generate edges to describe those objects to create a knowledge graph for searching. The program is called by
# all ranks in parallel so each rank only needs to work on a subset of the input. This allows us to use ML/DL to
# process large amounts of unstructured data at scale to create a searchable knowledge graph in seconds or minutes.
load_cmd = ['gen_image_edges.py',
            f"--images_dir=/container/oiv7/images",
            '--labels=/container/image_analysis/static_object_detection/coco_classes.pickle',
            '--model=frcnn-resnet',
            '--verbose=True']


obj_results = dsc.load(load_cmd, verbose=True)
```

 # Get all images that contains a “toaster”

```
select distinct ?img ?src
where {
   ?img a <urn://image> ;
     <urn://source> ?src ;
     <urn://contains> ?obj1 .
   ?obj1 a "toaster" .
}
```

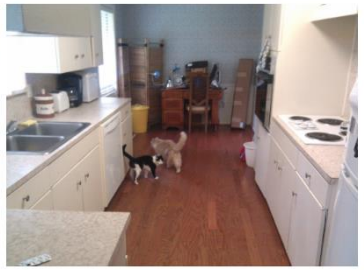

/tmp/files-17824-214034557089257/8467e13ae6baf890.jpg


/tmp/files-17824-214034557089257/02d749e0aeeaf231.jpg


/tmp/files-17824-214034557089257/0d48ba24443fb53a.jpg


/tmp/files-17824-214034557089257/d277ef1e35cb77b7.jpg


/tmp/files-17824-214034557089257/0d48ba24443fb53a.jpg


/tmp/files-17824-214034557089257/004ccb5669b648ab.jpg


/tmp/files-17824-214034557089257/0e08bca1d275efaa.jpg


/tmp/files-17824-214034557089257/5262d24ce30cb197.jpg

DEMO #1: Intelligence to tag/annotate/query using pre-trained AI models

Get all images with people and bus and a bicycle with confidence limits

```
select distinct ?img ?src
where {
   ?img a <urn://image> ;
     <urn://source> ?src ;
     <urn://contains> ?obj1 ;
     <urn://contains> ?obj2 ;
     <urn://contains> ?obj3 .
   ?obj1 a "person" ;
     <urn://confidence> ?con1 .
   ?obj2 a "bicycle" ;
     <urn://confidence> ?con2 .
   ?obj3 a "bus" ;
     <urn://confidence> ?con3 .
   filter(?con1 > .5 && ?con2 > .98 && ?con3 > .98)
}
```


/tmp/files-17824-214332385399699/2010d6d6c0474ae0.jpg


/tmp/files-17824-214332385399699/07cce5fbd1fe2ef9.jpg


/tmp/files-17824-214332385399699/008a2dd76102b748.jpg


/tmp/files-17824-214332385399699/2f024fe1406c0854.jpg

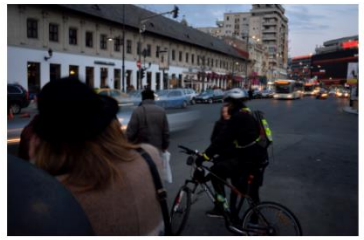

/tmp/files-17824-214332385399699/09bb5a893bbbba08.jpg


/tmp/files-17824-214332385399699/1cb5f33554f16f82.jpg


/tmp/files-17824-214332385399699/0308f52fb2be22d0.jpg


/tmp/files-17824-214332385399699/02ee45620889aa72.jpg


/tmp/files-17824-214332385399699/0606377016207be6.jpg


/tmp/files-17824-214332385399699/00f53e1960e157bb.jpg

DEMO #1: Query-time binding of pre-trained AI models

```
select distinct ?img ?src ?box1 ?box2
where {
  ?img a <urn://image> ;
    <urn://source> ?src ;
    <urn://contains> ?obj1 ;
    <urn://contains> ?obj2 ;
?obj1 a "person" ;
    <urn://bounding-box> ?box1 ;
    <urn://confidence> ?con1 .
  ?obj2 a "person" ;
    <urn://bounding-box> ?box2 ;
    <urn://confidence> ?con2 .
  filter(
        arq:user_func('predict_gender', ?src, '/container/models_age_gender_face', ?box1) = 'Male'
        && arq:user_func('predict_age', ?src, '/container/models_age_gender_face', ?box1) = '(25-32)'
        && arq:py_user_func('sentiment_udfs', 'get_sentiment', ?src, ?box1) = 'happy
        && arq:user_func('predict_gender', ?src, '/container/models_age_gender_face', ?box2) = 'Male'
        && arq:user_func('predict_age', ?src, '/container/models_age_gender_face', ?box2) = '(4-6)'
        && arq:py_user_func('sentiment_udfs', 'get_sentiment', ?src, ?box2) = 'happy')
}
```

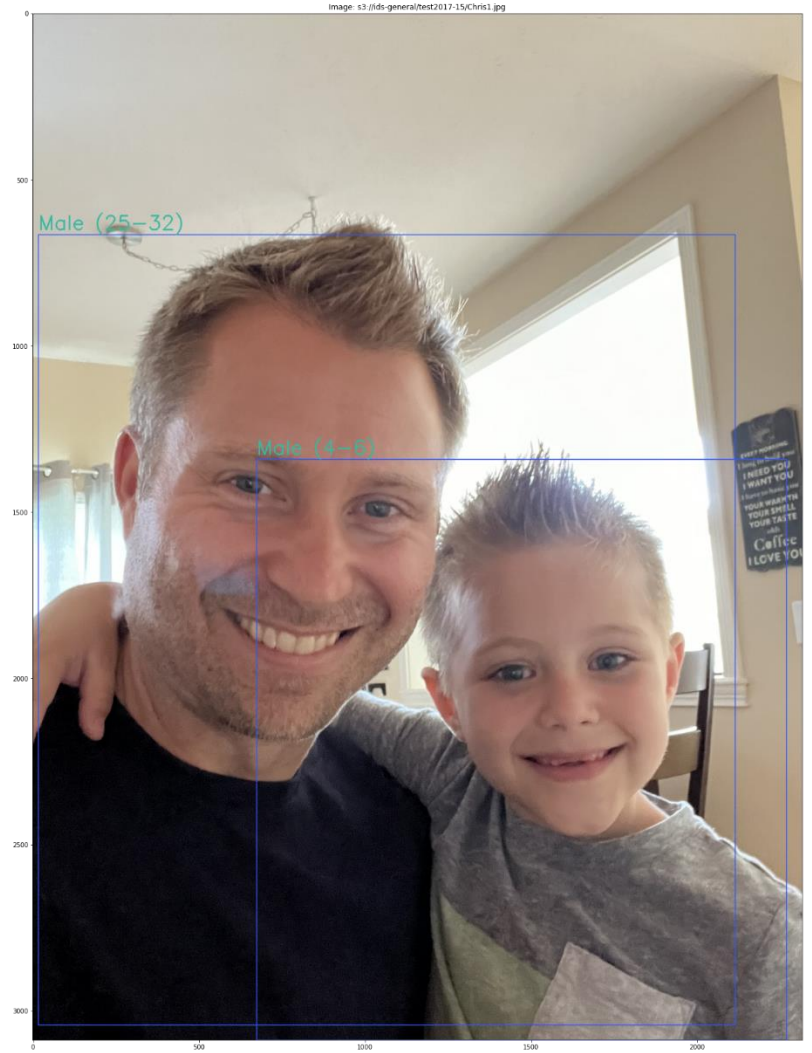
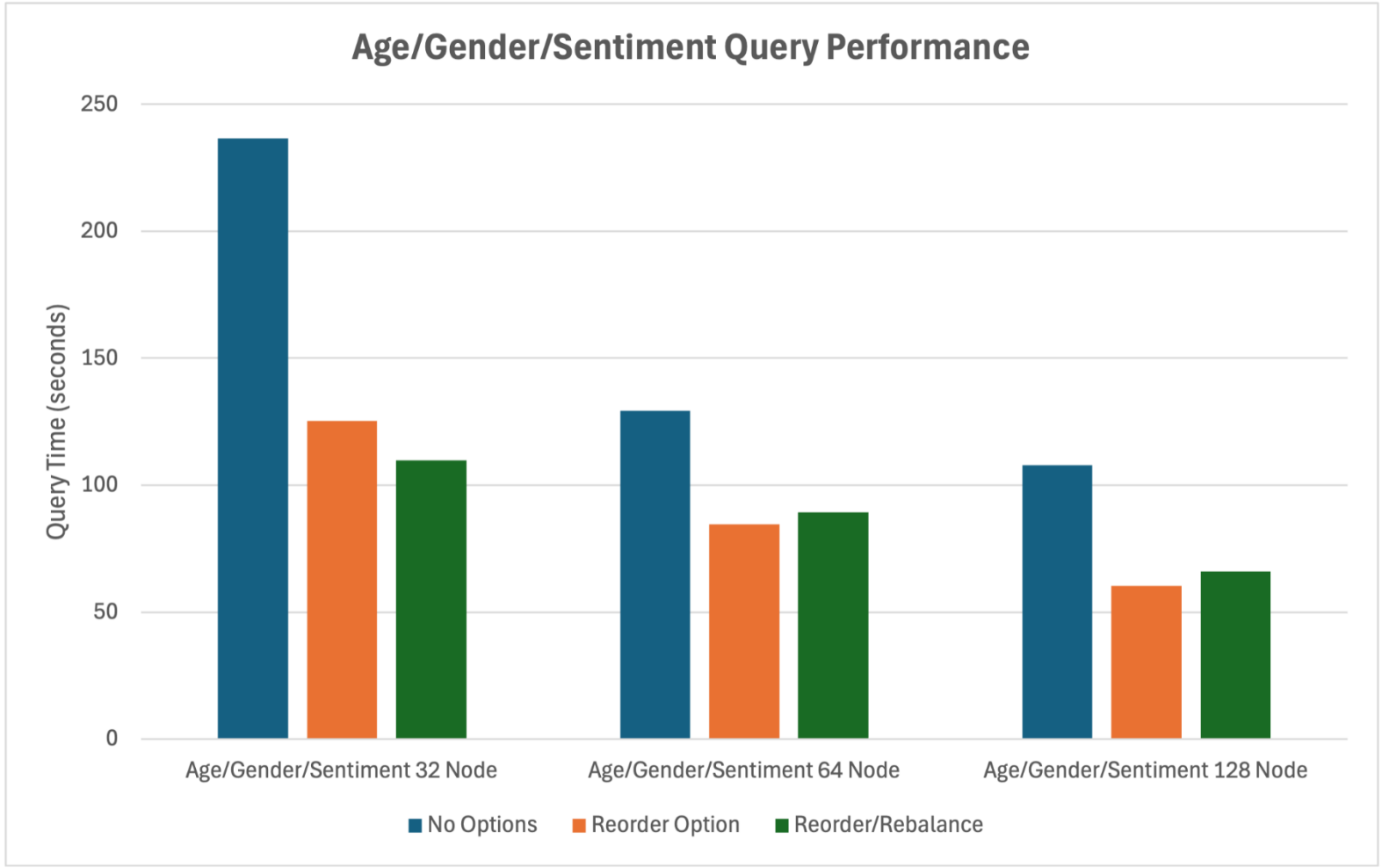


Image Analysis Query Performance

- Search 90K images
- > 1M potential solutions
  - 2 million – 12 million AI inferences per query
- Reordering and rebalancing both provide significant improvement
  - Query >2x faster at 32 nodes with both optimizations
  - ~40-45% reduction in query time on 128 nodes
- User cannot be expected to know best optimization
  - Automatically reordered so sentiment inferred first



DEMO #2: Drug discovery example on terabytes of data

```
Select <drug>
from <DataStore>
where
        <drug> is AI_generated (new_molecule)
                // Molecular GAN model
        <drug> has 3D structure
                // 3D structure prediction model
        <drug> docks with <protein> in 3D
                // 3D-3D alignment model
        <drug> interacts with <protein>
                // Ligand-Protein prediction
        <protein> is like <viral protein = COVID>
                // Protein similarity model
```

Query interface / APIs

Built-in analytics/algorithms

- **Clustering**
- **Ranking**
- **Spectral analysis**
- **Graph traversal**
- ....

User-defined AI

- **Molecular synthesis**
  - MolGAN, druGAN
- **Structure prediction**
  - AlphaFold, OpenFold
- **Protein-Ligand interaction**
  - DTBA, PharML
- **Sequence Similarity estimation**
  - Smith-Waterman algorithm

Data store

3D structures

Sequences

Time series

Associations

30 TBs on disk, 150+ billion medical facts

Can We Make Caffeine More Effective?

Uniprot
RCSB

AlphaFold

1. Adenosine A2A Receptor (Heterodimers and Heterotrimers)

Chembl 34

ChEMBL
BindingDB

2. Identify all natural compounds known to target (antagonize) 1 (e.g. caffeine)

DTBA
Autodock Vina

CB-Dock
SwissDock
SwissDrugDesign

3. Perform blind molecular docking at orthosteric and allosteric sites of A2A monomer, heterodimers, and heterotrimers

4. Orthosteric site: Identify new vasodilators

5. Allosteric site: Identify new down regulators of A2A activity (NAMs)

A. Structure similarity to compounds in 5 yields more A2A NAMs

B. NAM site similarity to A2A= more targets for compounds from 5

LOTUS
COCONUT
COllection of Open Natural Products

6. What plant contains hit compound(s) from 4?

8. What is the nutritional value of plant from 6?

Coconut

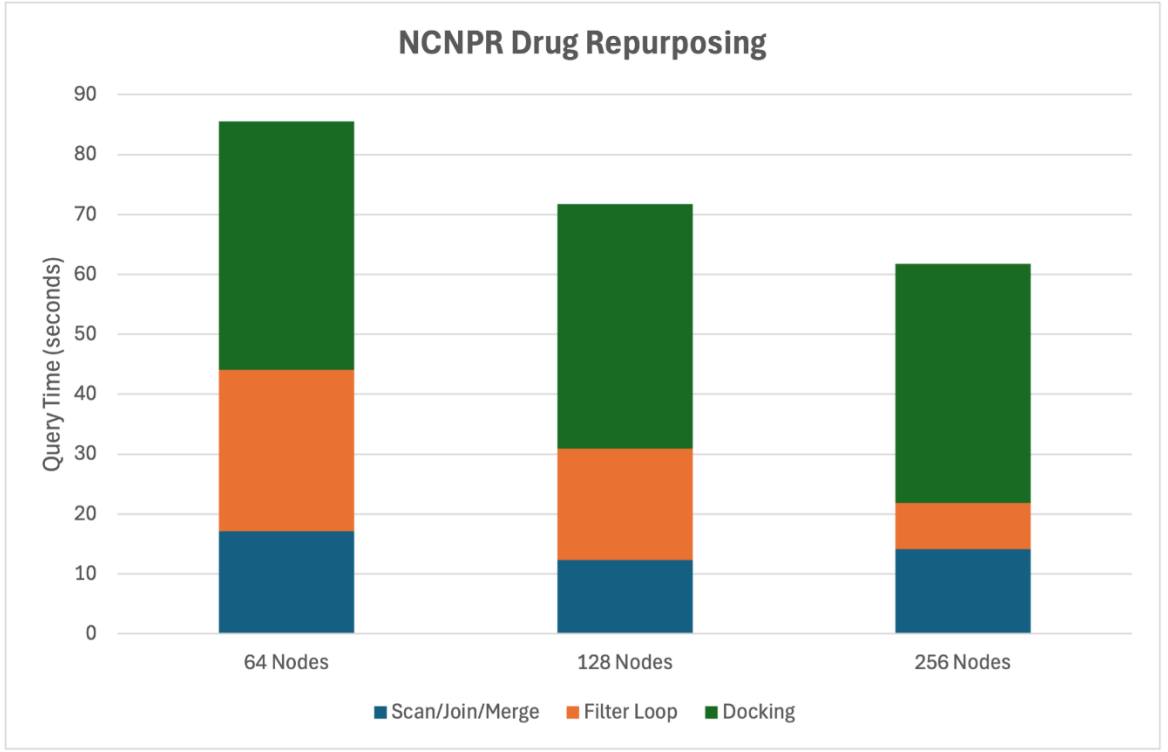
LOTUS
COCONUT
COllection of Open Natural Products

7. What plant contains hit compound(s) from 5?

9. What is the nutritional value of plant from 7?

NCNPR Drug Repurposing Query Performance

- Query steps:
  - Search for similar proteins
    - ~66M SW UDF operations
  - Search for compounds that target given proteins
  - Perform DTBA prediction
    - Hundreds or thousands of AI predictions
  - Perform docking simulations on candidate compounds
    - ~60 unique compounds to dock
- Predicts candidate compounds in seconds with docking
- Performance limited by slowest docking simulation
  - Times vary greatly per compound
  - Rest of the query scales well (scan/join/merge/filter)



Summary

IDS is a data intelligence platform that:

- **Hosts and serves data in different shapes**
  - 2D images, 3D point-clouds, genomic sequences, graphs, feature-vectors, vector embeddings, etc.
- **Orchestrates queries for exact, domain-specific, and AI search**
  - For database retrieval (exact search), pattern search using machine learning (approximate search), and user-defined functions (domain-specific search)
  - Dynamically optimizes queries based upon AI and domain-specific UDF profiling statistics
- **Conducts pattern search on the hosted data with query-time binding of AI models**
  - (e.g., pre-trained TensorFlow and PyTorch models)
- **Is a massively parallel processing database for unstructured data (fastest with graph data)**
  - Query latencies in seconds instead of minutes/hours
- **Is a "search engine" built on HPC first-principles**
  - Differentiated performance on a high-performance interconnect over commodity/cloud alternatives

Thank you

chris.rickett@hpe.com
sreenivas.sukumar@hpe.com
karlon.west@hpe.com