Big data serving: Processing and inference at scale in real time

By @jonbratseth

Big data serving will take over the *applied* part of *all* of applied AI

Big data maturity levels

Latent	Data is produced but not systematically leveraged	
	Example	**Logging**: Movie streaming events are logged.
Analysis	Data is used to inform decisions made by humans	
	Example	**Analytics**: Lists of popular movies are compiled to create curated recommendations for user segments.
Learning	Data is used to take decisions offline	
	Example	**Machine learning**: Lists of movie recommendations per user segment are automatically generated.
Acting	Automated data-driven decisions online	
	Examples	**Stream processing:** Each movie is assigned a quality score as they are added **Big data serving**: Personalized movie recommendations are computed when needed by that user.

Big data serving: Definition

Selection, **organization** and **machine-learned model inference**

- Over **many, constantly changing** data items (thousands to billions)
- With **low latency** (~100 ms) and **high load** (thousands of queries/second)

In short: ***AI + big data + online***

Advantages of using *big data*

Necessary in use cases like *search*, *recommendation* and many others, but

being able to consider relevant data always improves decision making

Intuition AI: Data compressed into a function (regression, ANN etc.)

Deliberate reasoning: Look up relevant data to make informed decisions

Just like humans, having “system 1” and “system 2”, AI need both

Advantages of making decisions *online*

Decisions use up to date information

Decisions are made *now*, and see the current state of the world

No wasted computation

Only those decisions that are needed will be made

Fine-grained decisions

A separate computation is made for each specific case

Architecturally simple

Just write data, and send the queries, in real time to the big data serving component

Big data serving: What is required?

Mutable state x distributed computing x low latency x high availability

Real-time actions: Find data and make inferences in tens of milliseconds.

Realtime knowledge: Handle data updates at a high continuous rate.

Scalable: Handle high requests rates over big data sets.

Always available: Recover from hardware failures without human intervention.

Online evolvable: Change schemas, logic, models, hardware while online.

Integrated: Data feeds from Hadoop, learned models from TensorFlow etc.

Introducing ...

vespa

Making big data serving universally available

Open source, available on https://vespa.ai (Apache 2.0 license)

Provenance:

- Web search: The canonical big data serving use case
- Yahoo search made Hadoop and Vespa to solve it
- Core idea of both: Move computation to data

Example usage: Vespa at Verizon Media (Yahoo)

TechCrunch, Huffington Post, Aol, Engadget, Gemini, Yahoo News, Yahoo Sports, Yahoo Finance, Yahoo Mail, etc.

Hundreds of Vespa applications,

… serving **over a billion** users

… **over 300.000** queries per second

… **over billions** of content items

… including

- Selecting and serving the personalized content of all landing pages and apps
- Selecting and serving the personalized ads on the world's 3rd largest ad network

Big data serving use case: Search

Data items: Text documents
Query: *Keywords*
Model(s) evaluated: *Relevance*
Selected items: *By relevance*

Vespa: Full text indexes, GBDT models, text match relevance features,
snippeting, linguistics, 2-phase ranking, text processing

"Search 2.0": Convert text to tensors and use vector similarity and neural nets

Vespa: Native *tensor* data model and computation engine, fast vector similarity
Coming soon: Approximate nearest neighbour search

Big data serving use case: Recommendation

Data items: Anything that can be recommended to somebody
Query: *Filters + user/context model*
Model(s) evaluated: *Recommendation*
Selected items: *By recommendation score*

Vespa: Native *tensor* data model and computation engine

Built-in support for models in TensorFlow, Onnx and XGBoost

Fast vector similarity search (*parallel WAND*)

Fast vector similarity brute force computation

Coming soon: Approximate nearest neighbour search (*with* filters)

Big data serving use case: Finance prediction

Data items: Assets (e.g stock)
Query: *World state update*
Model(s) evaluated: *Price predictor*
Selected items: *By largest price **change***

Result:

- Find the assets changing most in response to an event
- … using completely up-to-date information
- *… faster than anybody else*

Analytics vs big data serving

Analytics (e.g ElasticSearch)	VS	Big data serving (Vespa)
Response time in low seconds		Response time in low milliseconds
Low query rate		High query rate
Time series, append only		Random writes
Down time, data loss acceptable		HA, no data loss, online redistribution
Massive data sets (trillions of docs) are cheap		Massive data sets are more expensive
Analytics GUI integration		Machine learning integration

Where are we?

Done:

- Big data serving: What is it?
- Vespa: The big data serving engine

Remaining:

- Architecture: How are the *big data serving* challenges solved by Vespa?

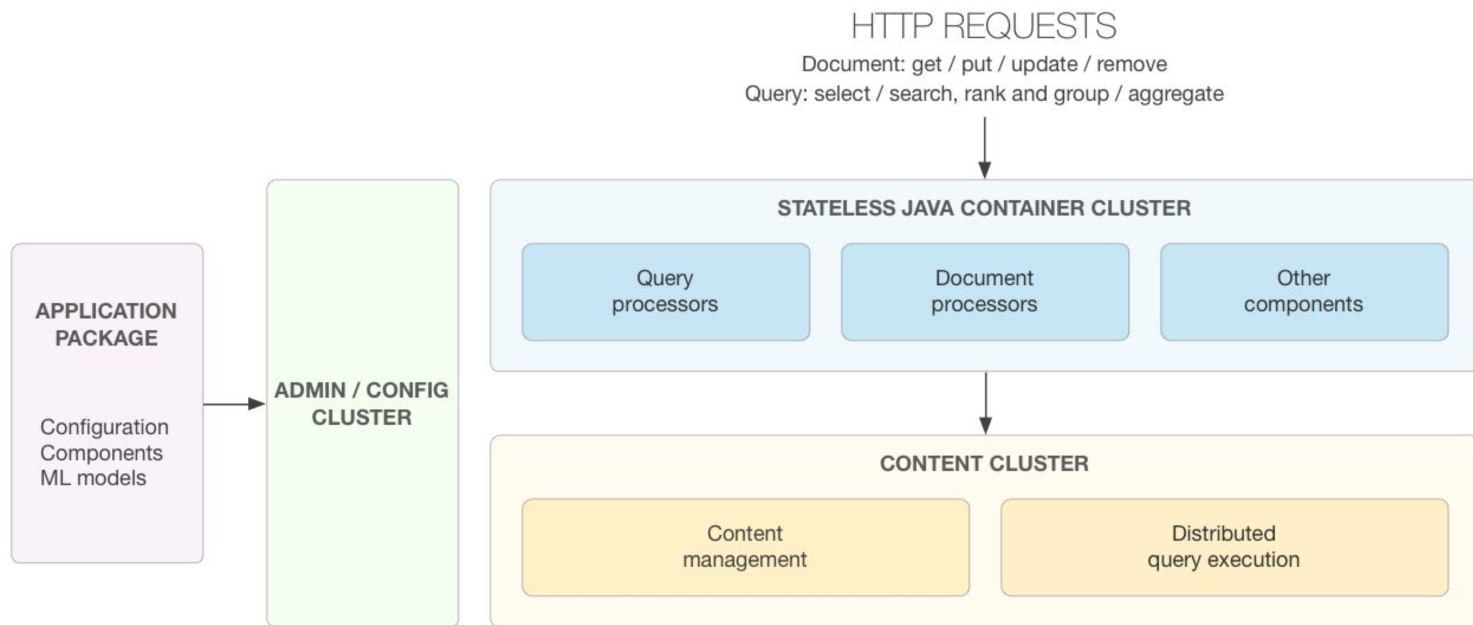
Vespa is

A platform for low latency computations over large, evolving data sets

- **Search** and **selection** over structured and unstructured data
- Scoring/relevance/inference: NL features, advanced ML models, TensorFlow etc.
- Query time **organization** and **aggregation** of matching data
- Real-time writes at a high sustained rate
- Live **elastic** and **auto-recovering** stateful content clusters
- Processing logic container (Java)
- Managed clusters: One to hundreds of nodes

Typical use cases: text search, personalization / recommendation / targeting, real-time data display, ++

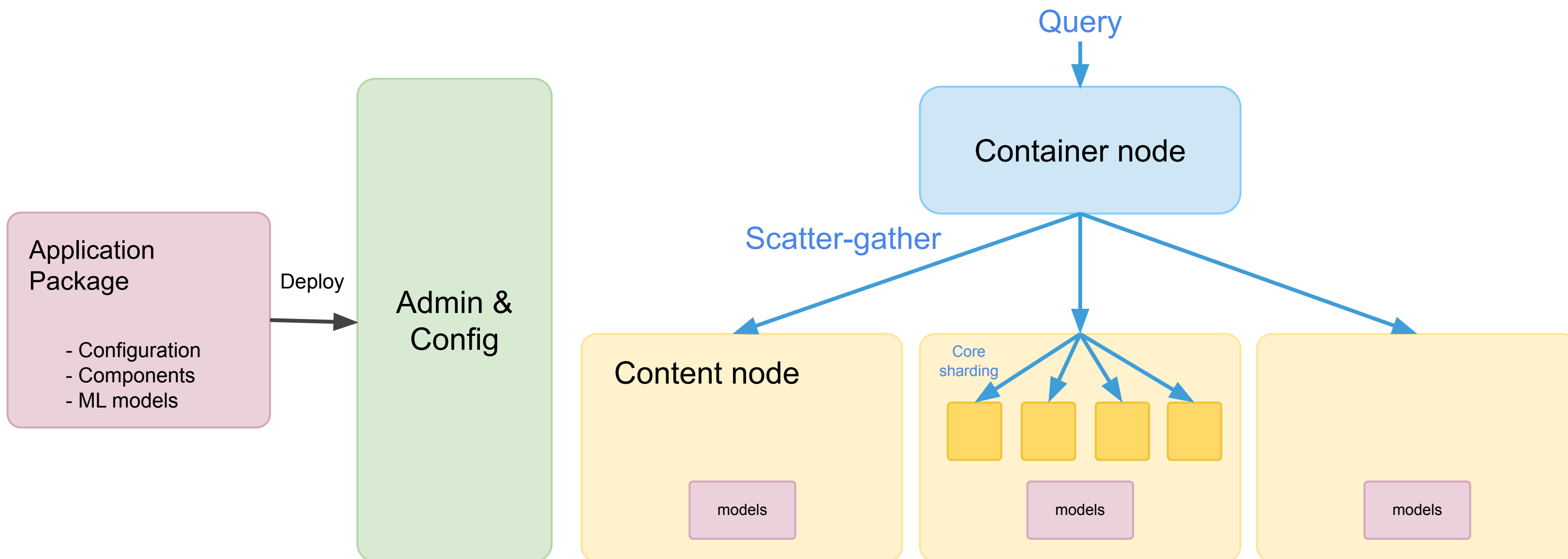
Vespa architecture



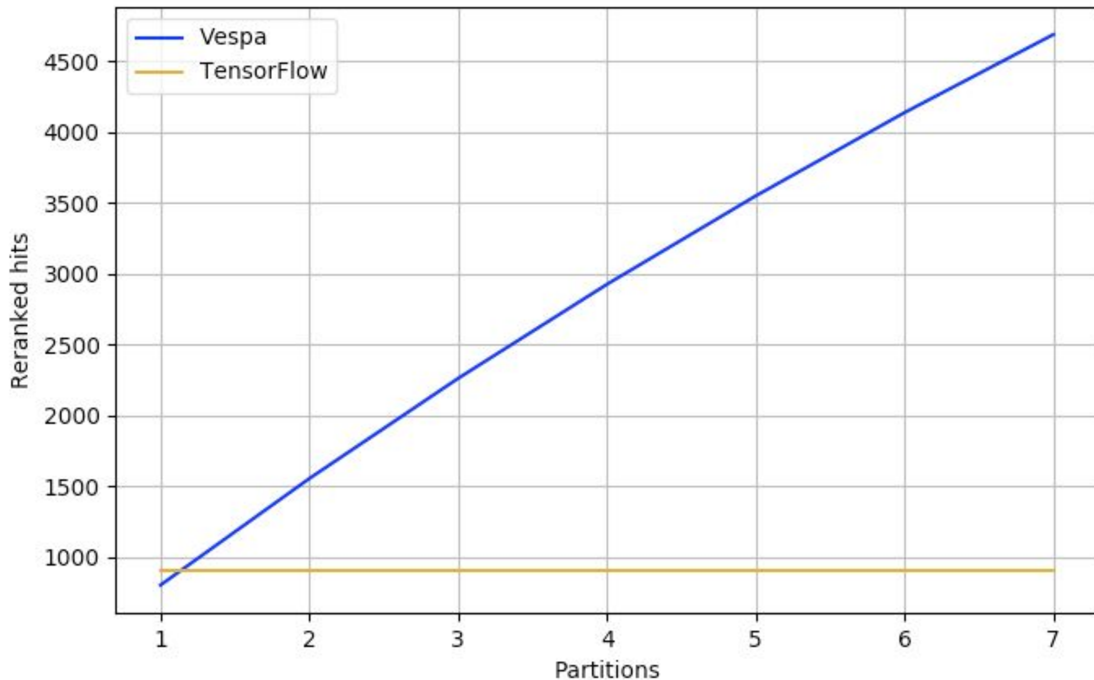
Scalable low latency execution:

How to bound latency in three easy steps

1) Parallelization
2) Prepare data structures at write time and in the background
3) Move execution to data nodes



Model evaluation: Increasing number of evaluated items



Latency: 100ms @ 95%
Throughput: 500 qps

10Gbps network

Takeaway:
Without distributing computation to data you run out of datacenter bandwidth surprisingly quickly

Query execution and data storage

- Document-at-a-time evaluation over all query operators
- *index* fields:
  - positional text indexes (dictionaries + posting lists), and
  - B-trees in memory containing recent changes
- *attribute* fields:
  - In-memory forward dense data, optionally with B-trees
  - For search, grouping and ranking
- Transaction log for persistence+replay
- Separate store of raw data for serving+recovery+redistribution
- One instance of all of this per doc schema

Data distribution

Vespa auto-distributes data over

- A set of nodes
- With a certain replication factor
- *Optionally*: In multiple node groups
- *Optionally*: With locality (e.g personal search)

Changes to nodes/configuration -> Online data redistribution

No need to manually partition data

Distribution based on CRUSH algorithm: Minimal data movement without registry

Inference in Vespa

Tensor data model: Multidimensional collections of numbers. In queries, documents, models

---

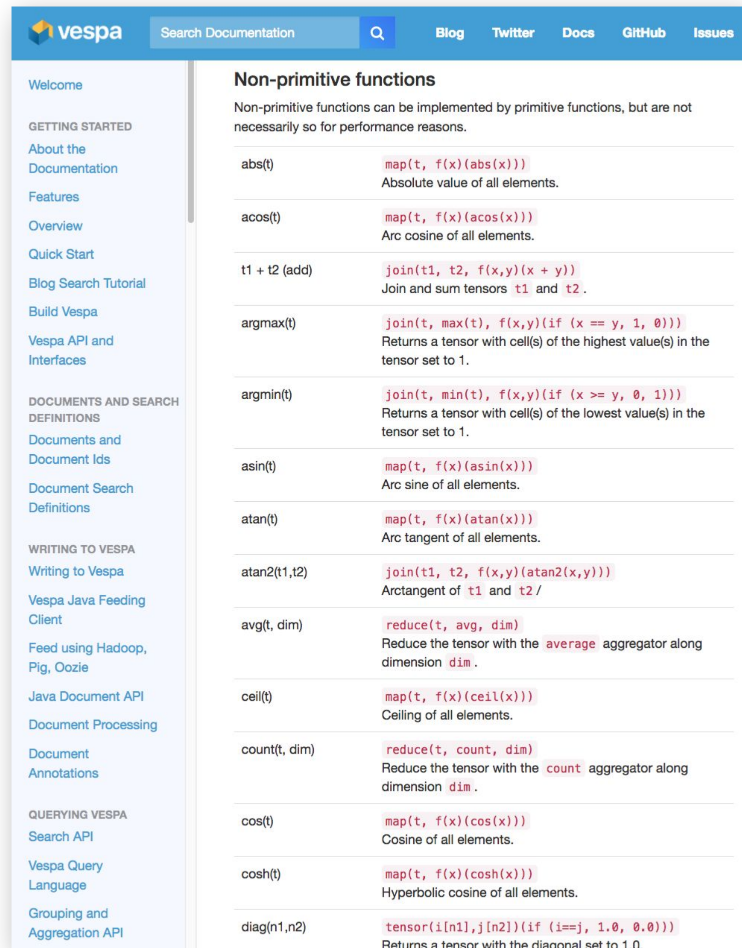
Tensor math express all common machine-learned models with join, map, reduce

---

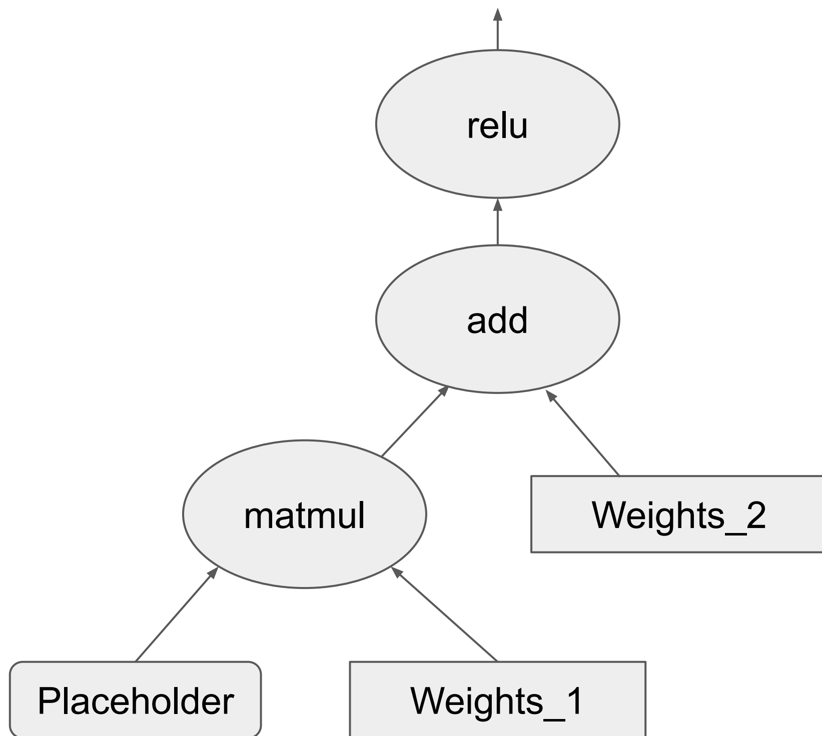
TensorFlow, ONNX and XGBoost integration: Deploy TensorFlow, ONNX (SciKit, Caffe2, PyTorch etc.) and XGBoost models directly on Vespa

---

Vespa execution engine optimized for repeated execution of models over many data items and running many inferences in parallel



Converting computational graphs to Vespa tensors



```
map(
    join(
        reduce(
            join(
                Placeholder,
                Weights_1,
                f(x,y)(x * y)
            ),
            sum,
            d1
        ),
        Weights_2,
        f(x,y)(x + y)
    ),
    f(x)(max(0,x))
)
```

Releases

New production releases of Vespa are made Monday to Thursday each week

Releases:

- Have passed our suite of ~1100 functional tests and ~75 performance tests
- Are already running the ~150 production applications in our cloud service

All development is in the open: https://github.com/vespa-engine/vespa

Recap

Making the best use of big data increasingly means **making decisions online**

Vespa is the **only** platform available for big data serving

---

Available at **https://vespa.ai**

Quick start: Clone and deploy your own app on our cloud at **https://cloud.vespa.ai**

Tutorial: Make a scalable blog search and recommendation engine from scratch
http://docs.vespa.ai/documentation/tutorials/blog-search.html

---

Twitter: **@vespaengine** **@jonbratseth**