TSVD 9

Technológie spracovania veľkých dát

Peter Bednár, Martin Sarnovský


APACHE
HBASE

Hbase

- Distribuovaná, škálovateľná stĺpcová NoSQL databáza postavená na HDFS
- Založená na Google Bigtable
- Určená pre spracovanie miliónov záznamov a stĺpcov
- Využíva výhod HDFS –fault tolerance a škálovateľnosť
- Časť Hadoop ekosystému pre čítanie/zápis do HDFS

Hbase - štruktúra

- Stĺpcovo orientovaná DB
- Schéma definuje tzv. Column families – hodnoty kľúč/hodnota
- Tabuľka môže mať viacero column families a každá z nich viacero stĺpcov
- Hbase štruktúra:
  - Tabuľka – kolekcia riadkov
  - Riadok – kolekcia columnfamilies
  - Column family – kolekcia stĺpcov
  - Stĺpec – kolekcia párov kľúč/hodnota

Rowid	Column Family			Column Family			Column Family			Column Family		
	col1	col2	col3	col1	col2	col3	col1	col2	col3	col1	col2	col3
1												
2												
3												

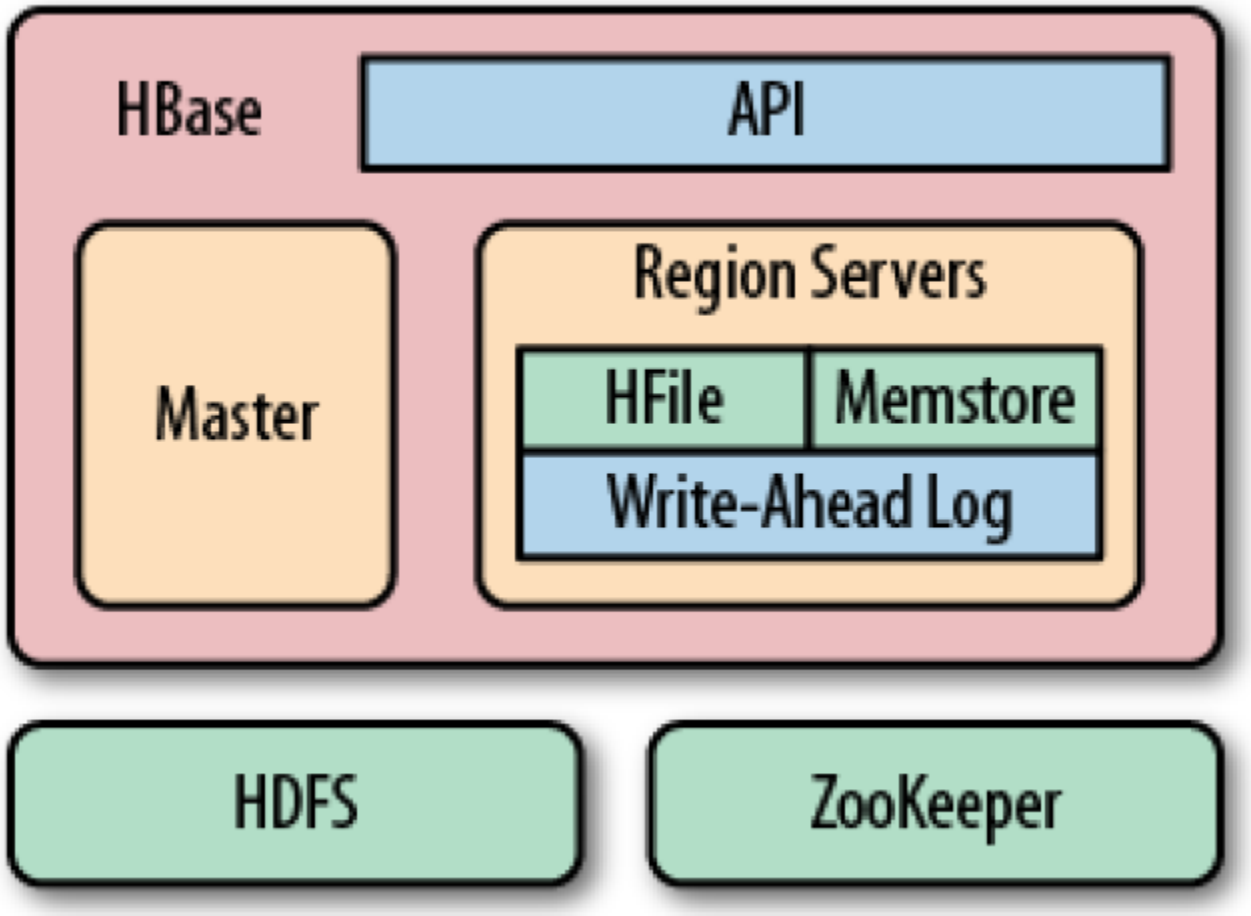
Hbase – príklad

ID riadku	Osobné dáta		Pracovné dáta	
ID zamestnanca	Meno	Mesto	Pozícia	Plat
1	Július	Poprad	Manager	4000
2	Karol	Košice	Developer	3000
3	Richard	Michalovce	Manager	4000

Hbase - vlastnosti

- Lineárne škálovateľná
- Automatická tolerancia zlyhania
- Dobrá integrácia s Hadoop platformou
- Replikácia dát naprieč klastrom
- Optimálizovaná pre dávkové spracovanie dát
- Java API pre klientský prístup
- Nepodporuje prístup a manipuláciu pomocou SQL.
  - Prístup aplikačne cez Java, REST, alebo ThriftAPI
  - Skriptovanie pomocou JRuby
- Nepodporuje operácie join
- Obmedzená podpora transakcií

Hbase - architektúra



Hbase

- Master
  - Monitoruje ostatné tzv. Region servery
  - Riadi a manažuje využívanie Region serverov
  - Smeruje klientov na správne Region servery
  - SPOF
- Regióny – tabuľky rozdelené a distribuované na Region servery
- Region servery
  - Spracovávajú a vykonávajú požiadavky (zápis/čítanie/prehliadanie) klientov na daných regiónoch
  - Posielajú Heartbeat hlavnému uzlu
  - Obsahuje samotné úložisko –memory store a Hfile
    - Memstore – ako cache (všetko, čo vstúpi do Hbase sa najprv uloží tu)
    - Hfile – dáta z cache transformované na bloky a uložené
- Zookeeper–manažuje prostredie


HIVE

Hive

- Technológia vyvinutá Facebook-om, ktorá pridáva do Hadoop prostredia funkcionalitu dátového skladu dopytovacie možnosti plne kompatibilné s dopytovacím dialektom SQL
- Hive je nástroj infraštruktúry dátového skladu pre spracovanie štruktúrovaných dát v Hadoop prostredí = infraštruktúra- vybudovaná nad Hadoop prostredím
- Dôvod vzniku - častokrát veľmi zložité písanie aplikácií priamo pomocou paradigmy MapReduce
- Poskytuje sumarizáciu, dopytovania a analýzy dát, mechanizmus pre uloženie štruktúry rôznym dátovým formátom, prístup k dátam priamo v HDFS a vykonávanie dopytov pomocou MapReduce
- Poskytuje SQL rozhranie pre prístup k štruktúrovaným dátam, ktoré sú uložené v HDFS

Hive

- Hive nie je
  - Relačná databáza
  - On-Line Transaction Processing (OLTP) nástroj
  - Nie je určený pre real-time analýzy a dopytovanie
- Vlastnosti Hive
  - Ukladá schému v databáze a spracovávané dáta v HDFS
  - Navrhnutý pre OLAP
  - Poskytuje dotazovanie pomocou SQL-type jazyka HiveQL alebo HQL
  - Rýchly, škálovateľný a rozšíriteľný

Hive

- Nie je vhodný pre transakčné spracovanie, alebo pre dopytovanie v reálnom čase
- Vhodný pre úlohy na veľkých súboroch dát a úlohy dávkového typu - vďaka svojej škálovateľnosti na veľké klastre, podpore rozšírení v jazyku HiveQL
- Dopyty vytvorené v HiveQL sú potom pri vykonaní kompilátorom preložené na orientované acyklické grafy skladajúce sa z MapReduce úloh
- Úlohy sú potom vykonané pomocou YARN alebo MapReduce 1 (podľa používanej verzie Hadoop)
- Hive štruktúruje dáta do dobre pochopiteľných konceptov ako napr. tabuľky, stĺpce, riadky, partície atď.

Hive – dátový model

- Tabuľky
  - Analógia k tabuľkám v relačných DB
  - Každá tabuľka má korešpondujúci adresár v HDFS
- Partície
  - Vnorené pod-adresáre v HDFS pre zodpovedajúce kombinácie stĺpcov
- Buckets (Clusters)
  - Pre účely paralelizácie – rozdelené dáta podľa hashovacej tabuľky
  - Dáta v jednej partícii rozdelené do buckets podľa hashovacej funkcie
- Externé tabuľky
  - Odkazujú na existujúce dáta do HDFS

Hive – dátový model

- Podpora dátových typov:
  - Integer, Boolean, Float, String
- Komplexné dátové typy
  - Štruktúry
  - Mapy – páry kľúč/hodnota
  - Polia – indexované zoznamy

Hive – architektúra

- UI – sprostredkuje rozhranie s používateľom, napr. Hive Command Line, Web UI
- MetaStore – Hive používa databázový server pre ukladanie schémy, metadáta o tabuľkách atď., a mapovanie na HDFS
- HiveQL Process Engine – písanie SQL dotazov a ich MapReduce vykonávanie
- Execution Engine – spracováva a vykonáva dotazy, využíva MapReduce
- Úložisko - HDFS, alebo Hbase, používajú sa na ukladanie samotných dát

Pig

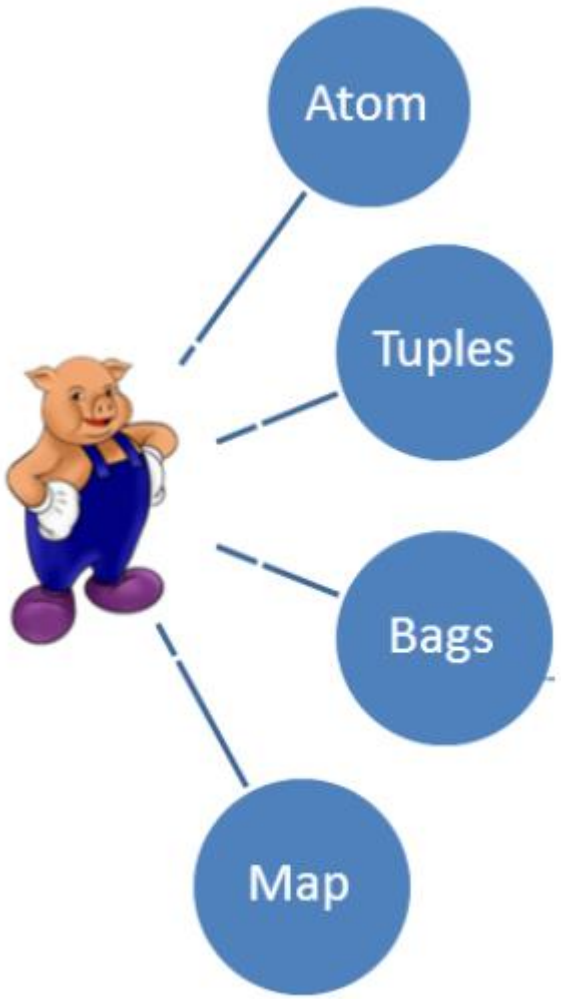
- Vyvinutý v Yahoo, neskôr presunutý do projektu Hadoop ako jeho súčasť
- MapReduce
  - nepodporuje komplexné viackrokové dátové prúdy
  - nepodporuje kombinované spracovanie viacerých súborov dát
  - problém písania každej jednoduchej operácie znova a znova, to vytvára miesto pre vznik chýba zbytočne zvyšuje trvanie analýzy dát
- Tieto nedostatky => *Pig*, systém na písanie komplexných MapReduce transformácií použitím jednoduchého skriptovacieho jazyka (*Pig Latin*)
- Skladá sa z 3 základných častí
  - Pig Latin programovacieho jazyka,
  - prostredia pre vykonávanie programov Pig
  - repozitára používateľom definovaných funkcií

Pig

- Vlastnosti
  - Vyjadruje sekvencie MapReduce úloh
  - Dátový model: vnorené “bags” položiek
  - Poskytuje relačné (SQL) operácie (JOIN, GROUP BY, atď.)
  - Možnosť použiť Java funkcie
- Rýchlejší vývoj aplikácií v porovnaní s MapReduce
- Zameranie viac na dataflow ako na logiku aplikácie
- Zameriava sa na aspekty spájania úloh a dátový tok

Pig

- Každá úloha Pig skriptu je prekladaná do jednej alebo viacerých MapReduce úloh
- Preklad – rozbor úlohy –pri rozbore sa kontroluje správnosť syntaxe, kontrola definovania premenných, kontrola typov
- Výstup z rozboruje kanonický logický plán, ktorý priamo korešponduje s údajmi pôvodného programu a je vo forme orientovaného acyklického grafu, ako je tomu aj v prípade Hive



- Pig Latin používa nasledujúce dátové typy
  - Atom
    - Atomické dátové hodnoty napr. ‘Hadoop’ alebo ‘1’
  - N-tice (tuples)
    - Záznam, ktorý pozostáva zo sekvencie (ako záznamy v SQL) napr. (‘Hadoop’,’Pig’)
  - Data Bag
    - Množina n-tíc – ekvivalent tabuľky v SQL terminológii napr. {(‘Hadoop’, ‘Pig’), (‘Hadoop’, (‘HDFS’, ‘Hive’))}
  - Data Map
    - Asociatívne pole, množina párov kľúč/hodnota, napr.

Pig-Latin

- Procedurálny jazyk pre vyjadrenie tokov úloh analýzydát
- Zameraný viac na to čo chce používateľ vykonať, ako na to, ako to bude vykonané
- Programovanie je podobné špecifikácií pracovného toku
- Umožňuje programátorovi explicitne kontrolovať tok dát v úlohe na spracovanie dát
  - Písanie skriptov, ktoré reťazia MapReduce úlohy
- Podporuje rozšírenie pomocou používateľom definovaných funkcií, ktoré môžu byť v rôznych jazykoch
- Samotný program sa skladá z troch základných častí
  - umiestnenie požadovaných dát
  - transformácie dát, ktoré budú neskôr preložené na map() a reduce() úlohy
  - kam budú výsledky nasmerované

Pig - príkazy

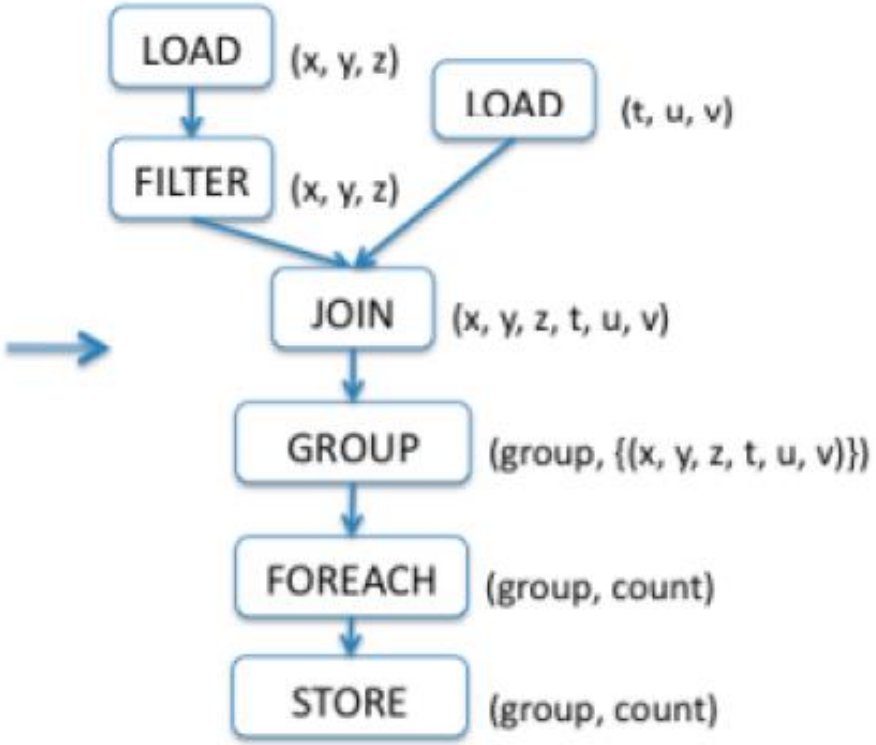
- load – číta dáta zo súborového systému
- store – zapisuje dáta do súborového systému
- foreach – aplikuje výraz pre každý záznam a vyprodukuje jeden alebo viacero záznamov
- filter – odstráni záznamy, pre ktoré nie je splnená podmienka (nevráti true)
- group/cogroup – zoskupuje záznamy podľa zadaného kľúča
- join – spojí 2 a viac vstupov na základe kľúča. Viacero joining algoritmov v dostupných
- order – zoradí záznamy podľa kľúča
- distinct – odstraní duplikáty
- union – spojí 2 datasety
- split – rozdelí dáta do 2 a viac podmnožín na základe podmienky
- stream – Send all records through a user provided executable
- sample – načíta náhodnú vzorku dát
- limit – obmedzí počet záznamov

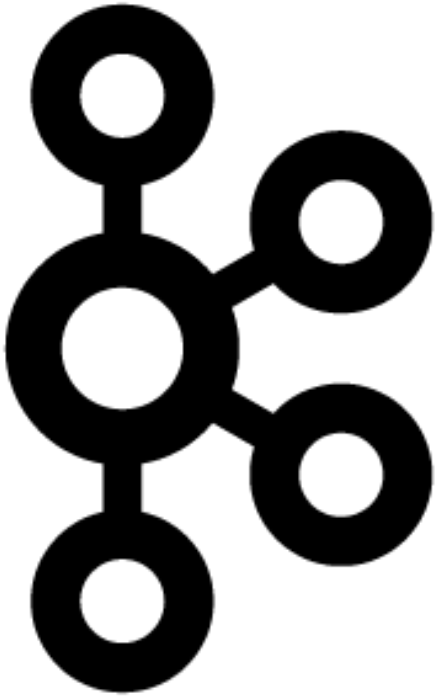
Pig - príklad

Pig Latin

```
A = LOAD 'file1' AS (x, y, z);
B = LOAD 'file2' AS (t, u, v);
C = FILTER A by y > 0;
D = JOIN C BY x, B BY u;
E = GROUP D BY z;
F = FOREACH E GENERATE
    group, COUNT(D);
STORE F INTO 'output';
```

Logical Plan





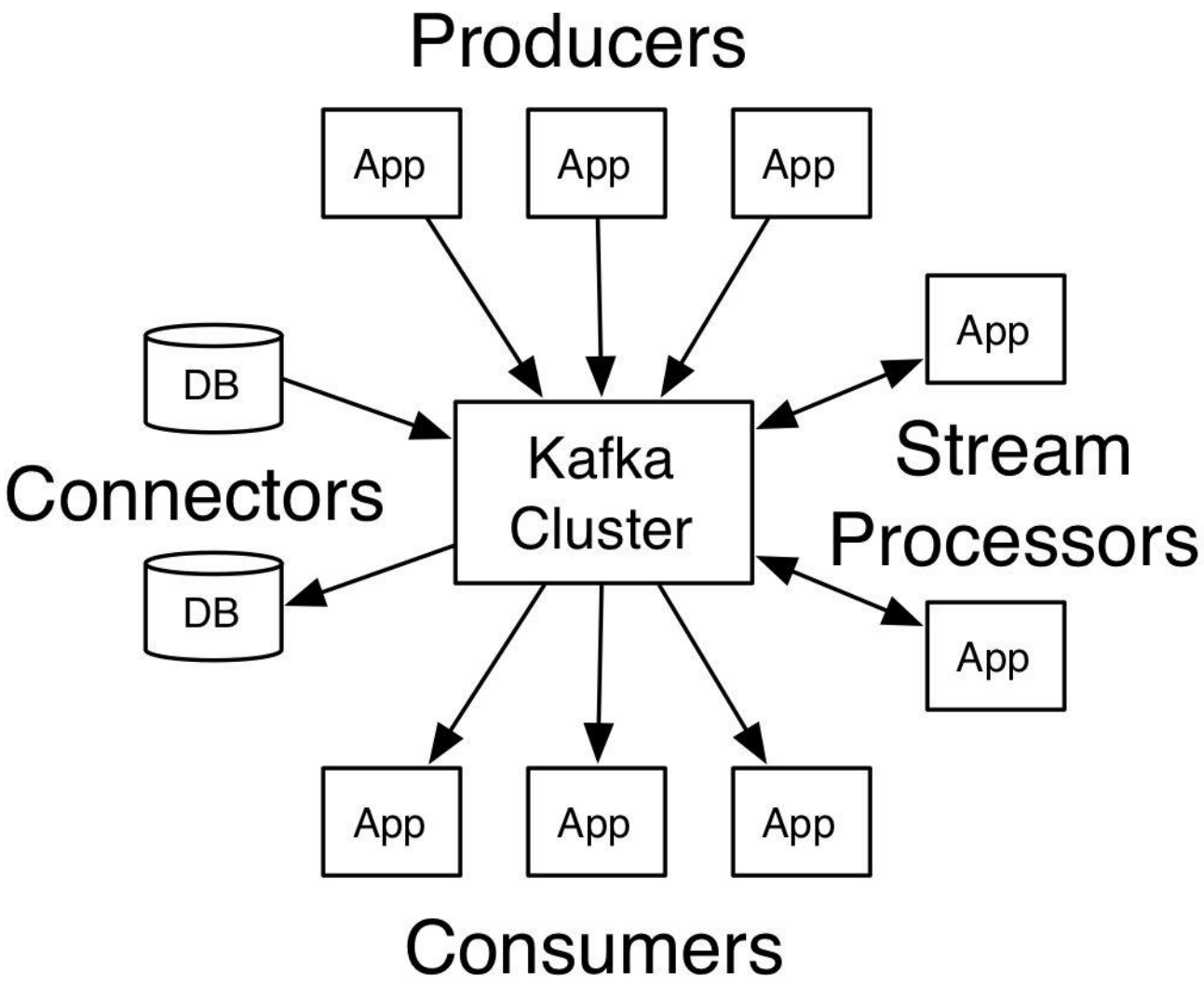
kafka

Kafka

- Apache Kafka – distribuovaná subscribe-publish služba pre zasielanie správ
- Pôvodne vznikla v LinkedIn, cez Apache Incubator se dostala medzi hlavné projekty Apache Software Foundation
- Kafka pracuje s prúdmi správ rozdelených do kategórií, ktoré nazýva témami (topics)
- Proces, ktorý produkuje správy – Producent (Message producer)
- Konzument – proces, ktorý odoberá a spracováva správy podľa špecifikovanej témy
- Kafka uchováva všetky publikované správy, nezávisle na tom, či boli alebo neboli konzumované (podľa nastaveného úseku)

Kafka

- Metadáta pre konzumenta – uchovávané v logu správy, obsahujú údaje o pozícii konzumenta
- Log rozdelený do častí – distribuované naprieč servermi v klastri
- Každá časť logu je replikovaná - kvôli tolerancii chýb
- Každá táto časť má server, ktorý pre ňu vystupuje ako Leader, ostatné sú nasledovníci
- Každý server teda vystupuje ako Leader pre niektoré časti logu a zároveň ako nasledovník pre iné


Producers
App
App
App
App
Stream
Processors
App
App
DB
DB
Connectors
Kafka
Cluster
App
App
App
Consumers

Kafka

- Vlastnosti
  - vysoká priepustnosť,
  - nízka latencia,
  - masívna škálovateľnosť
- Distribuovanosť – časti logu
- Využitie
  - ako zbernica správ, rozhranie pre monitoring, sledovanie dát o používateľoch, agregácia logov, atď.

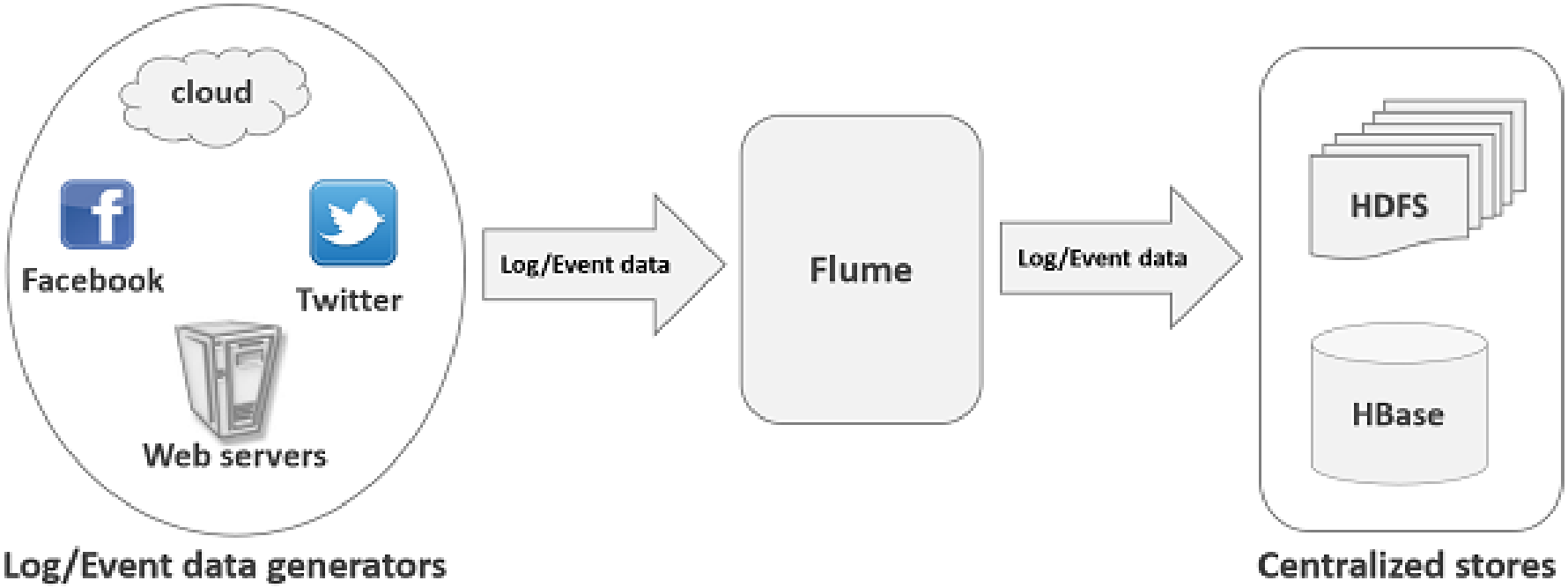

FLUME

Apache Flume

- Distribuovaná služba pre kolektovanie dát vo forme logov alebo udalostí
- Zbiera dátové prúdy logov zo zdrojov a agreguje ich pre ďalšie spracovanie
- Vlastnosti
  - Spoľahlivosť
  - Škálovateľnosť

Flume – dátový model

- Tok (flow) – zodpovedá typu dátového zdroja (logy zo servrov, metriky z monitoringu)
- Toky pozostávajú zo zreťazených uzlov
- Uzol (node) – v uzloch sú dáta buď spracované dekorátorom (decorator) a potom exportované pomocou sink
  - Decorator – sampling, kompresia, projekcie, extrakcia
  - Sink – konzola, lokálny FS, HDFS, ostatné uzly


cloud
Facebook
Twitter
Web servers
Log/Event data generators
Log/Event data
Flume
Log/Event data
HDFS
HBase
Centralized stores

Apache Storm

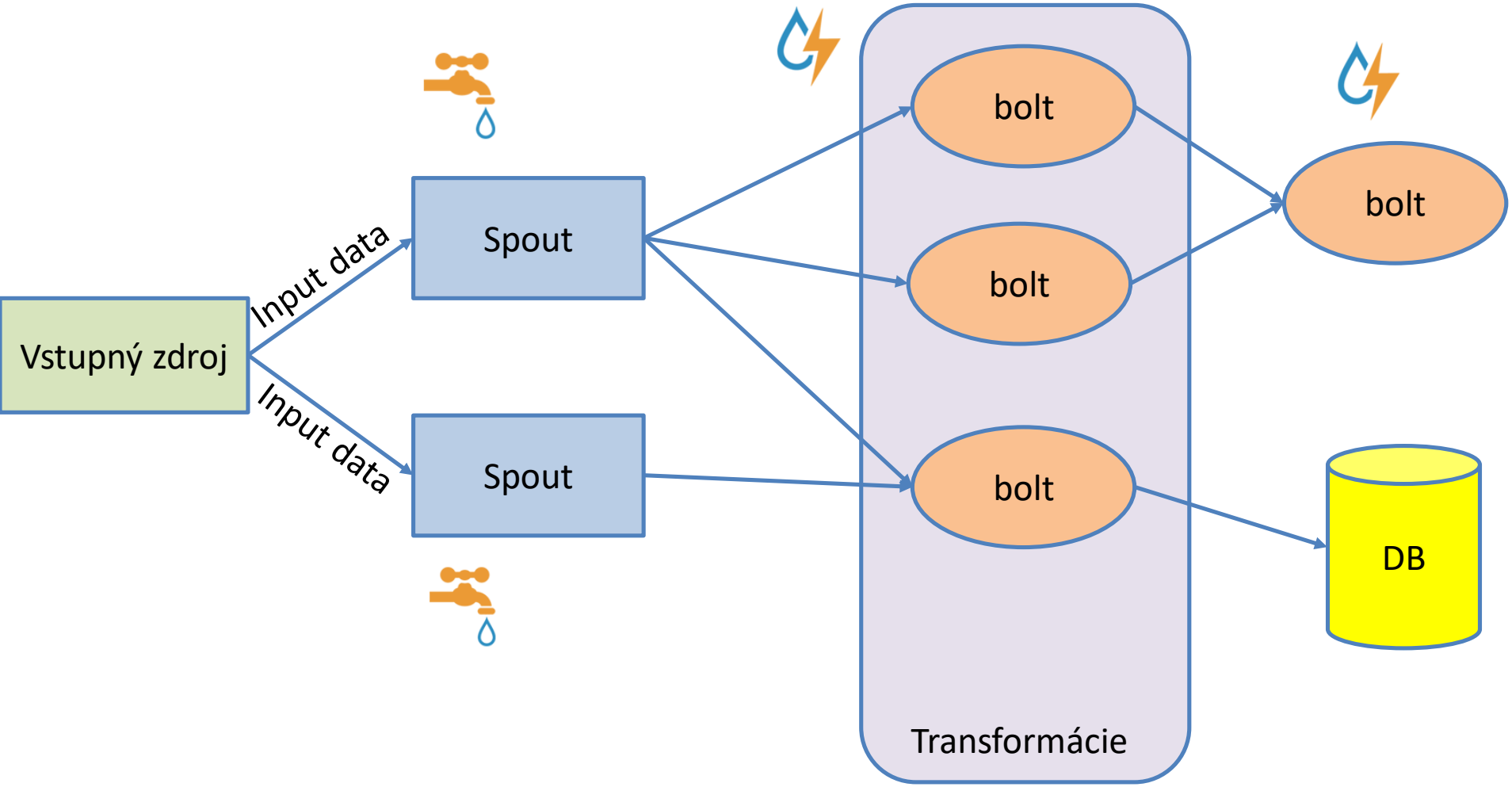
- Distribuovaný systém pre spracovanie prúdov dát v reálnom čase
  - Distribuovaný – škálovateľnosť pridávaním ďalších zariadení do klastra
  - Spoľahlivý – zaručenie, že každá správa sa doručí aspoň raz
  - Odolný voči zlyhaniu
- Používa sa pre
  - spracovanie prúdov dát
  - nepretržité počítanie (continuous computation)
  - Distribuované RPC

Storm komponenty

- Komponenty Storm sa delia do dvoch skupín z hľadiska charakteru činnosti
- Spouts – vstupné komponenty predstavujú zdroje prúdov dát a preposielajú do komponentov typu Bolt
- Bolts – vykonávajú transformácie získaných dát; tie potom buď ukladajú, alebo preposielajú na iné Bolt komponenty

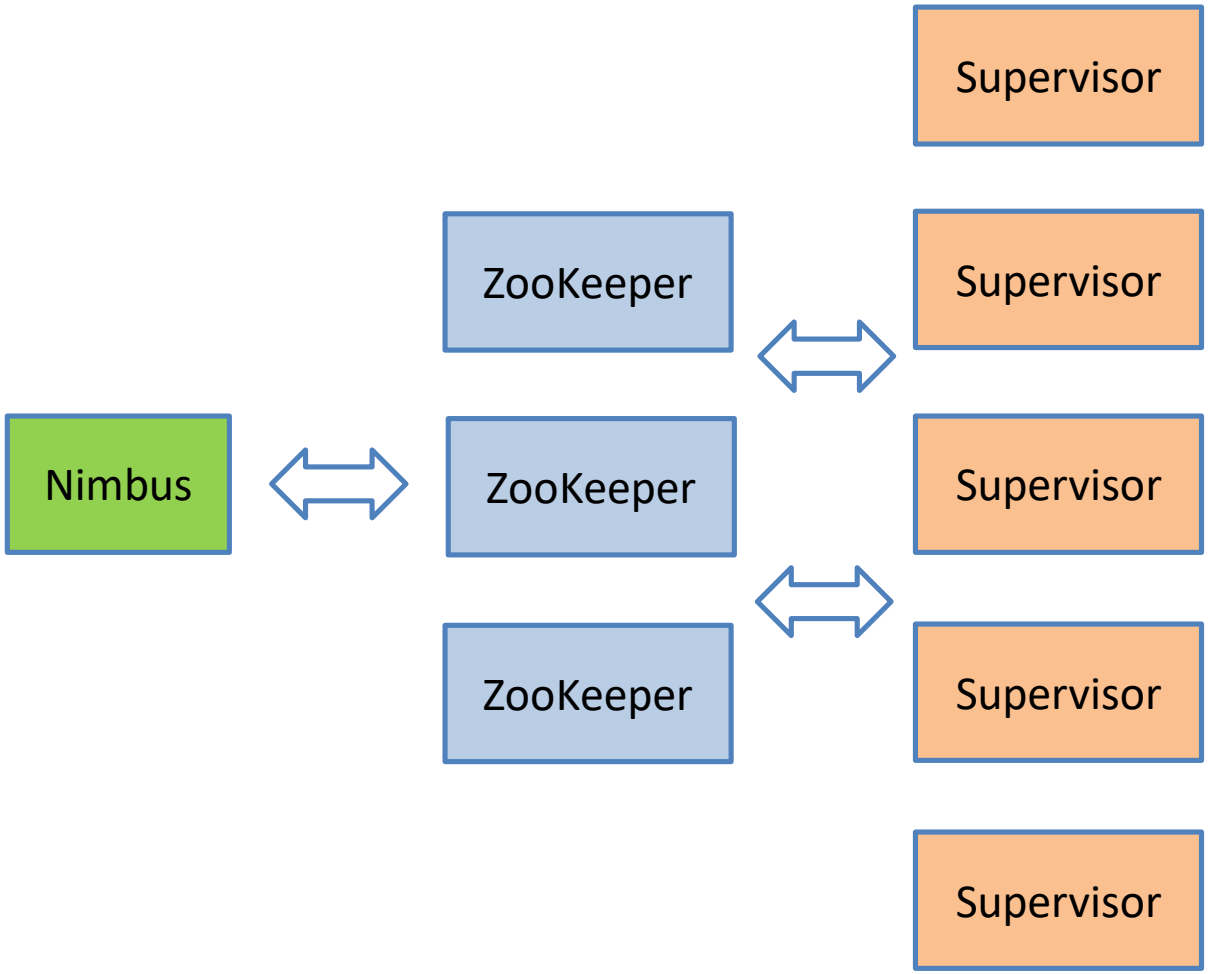
Komponenty Storm

- Tuples – základná dátová jednotka, usporiadaná n-tica elementov, napr. (7,3,2,1), posielajú sa medzi jednotlivými komponentami
- Stream – prúd, neohraničení sekvencia n-tíc
- Spout – zdroj dátového prúdu (napr. Twitter API)
- Bolts – spracuje vstupujúci prúd a vyprodukuje výstupný prúd, môže transformovať prúd (funkcie, filter, agregácia, join atď.)
- Topológia – celkový výpočet, reprezentovaný ako orientovaný necyklický graf skladajúci sa z uzlov (Spouts a Bolts), hrany dátové toky


Vstupný zdroj
Input data
Input data
Spout
Spout
bolt
bolt
bolt
bolt
DB
Transformácie

Storm architektúra

- Storm klaster pozostáva z 3 typov uzlov
- 1. Nimbus (master, podobný JobTrackeru z Hadoop architektúry)
  - Jeden na klaster, distribuuje kód do klastra, distribuuje dáta naprieč klastrom, spúšťa pracovné uzly, monitoruje výpočty a re-alokuje pracovné uzly v prípade potreby
- 2. Zookeeper
  - Koordinuje zdroje v Storm klastri, zachytáva a sleduje stavy uzlov
- 3. Supervisor
  - Beží na pracovných uzloch, realizuje výpočtovú časť topológie, komunikuje s Nimbus uzlom cez Zookeeper, štartuje a zastavuje pracovné uzly na základe pokynov z Nimbus uzla


Supervisor
ZooKeeper
Supervisor
Nimbus
ZooKeeper
Supervisor
ZooKeeper
Supervisor
Supervisor

Paralelizmus v Storm

- 3 druhy entít
- Worker
  - Vykonáva podmnožinu topológie, môže spustiť jeden alebo viac exekútorov pre jeden alebo viac komponentov (Spouts/Bolts)
- Exekútor
  - Vlákno Workera, ktoré sa vykonáva v JVM Workera, jeden exekútor môže spustiť jednu alebo viac úloh, ale vždy pre jeden komponent (Spout/Bolt), úlohy bežia sériovo
- Úloha
  - Predstavuje skutočné spracovanie dát, každá inštancia Spoutu/Boltu je realizovaná jednou úlohou

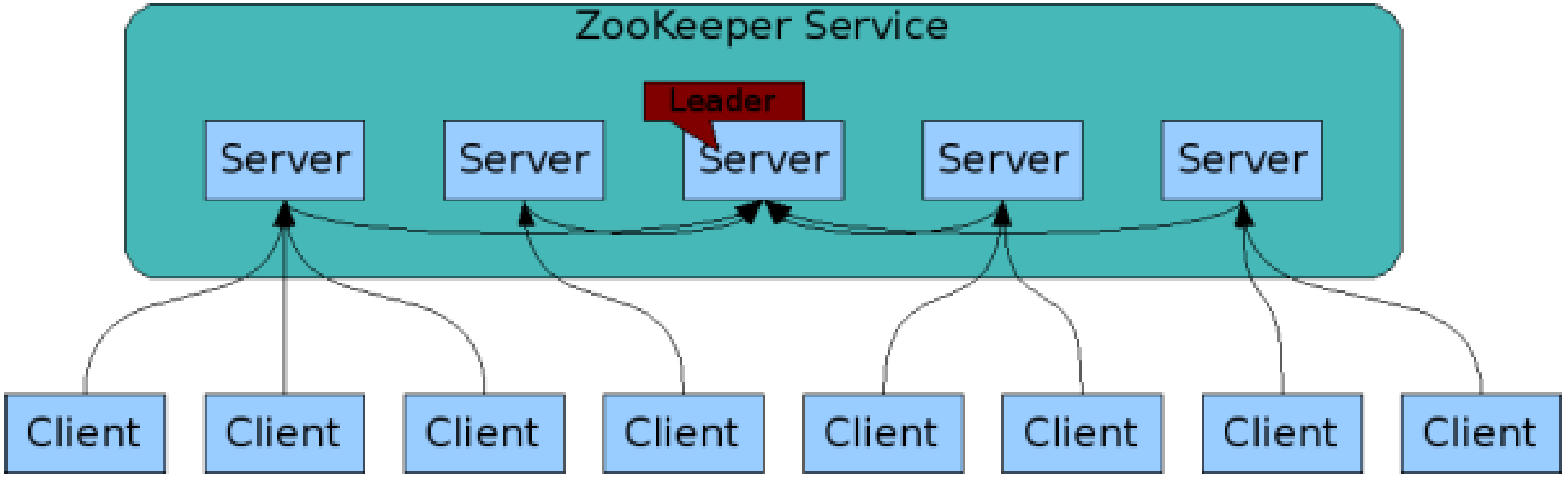

ZOO

ZooKeeper

- Centralizovaná služba pre konfiguračný manažment a synchronizáciu distribuovaných prostredí
- Koordinuje distribuované procesy cez zdieľaný namespace registrov (znodes) – dátový model podobný súborovému systému
- Služba potom poskytuje klientom prístup k znodes (s nízkou latenciou a vysokou dostupnosťou)
- Každý znode potom má asociované dáta
- Dáta – konfigurácia, info o lokácii, stav uzlov atď.

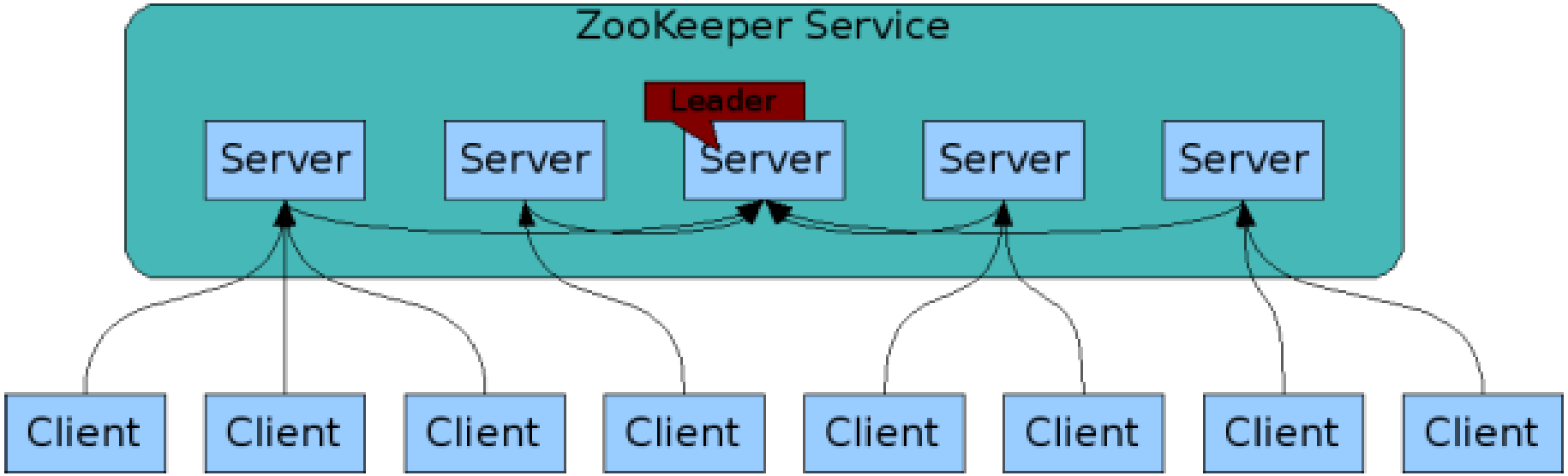
ZooKeeper

- ZooKeeper – distribuovaná služba, replikovaná na viacerých uzloch klastra
- Jeden zo ZooKeeper uzlov je zvolený do role Leader-a
- Hlavnou úlohou Leader-a – zoradiť požiadavky klientov, ktorí menia stav (napr. zapisujú dáta od klientov)



ZooKeeper

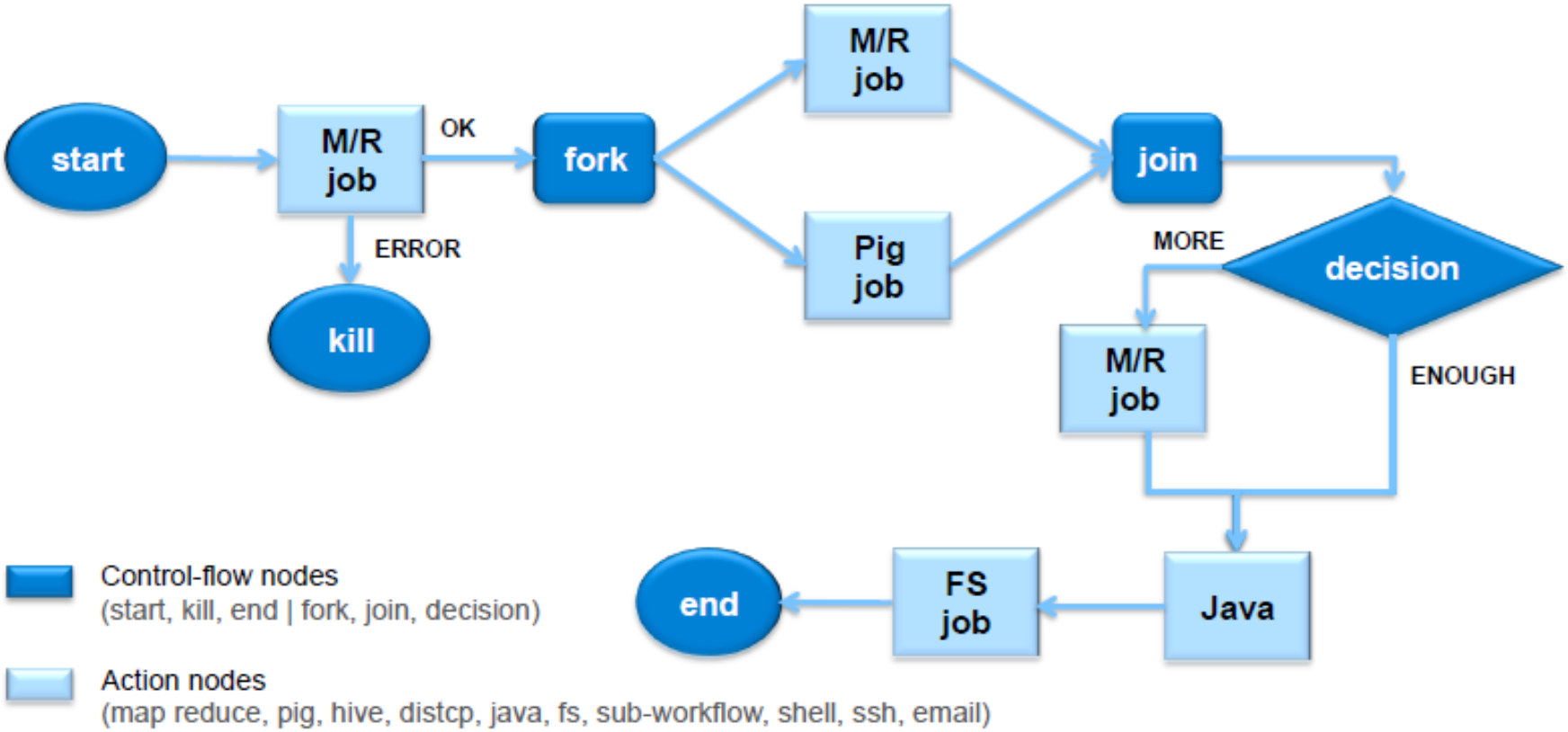
- Každý ZooKeeper server uchováva znodes hierarchiu v pamäti (kvôli nízkej latencii)
- Klient sa pripája na jeden zo ZooKeeper serverov a udržiava s ním konektivitu
- ZooKeeper služba je dostupná, keď je dostupná väčšina ZooKeeper serverov




OOZIE

Oozie

- Plánovací systém pracovného toku pre úlohy systému Hadoop
- Integrovaný s Hadoop prostredím, podporuje úlohy MapReduce, Pig, Hive, Sqoop a mnoho ďalších
- Oozie sa skladá s dvoch typov úloh
  - Action uzly
  - Control flow uzly
- Definícia a spúšťanie pracovného toku
  - kolekciu akcií usporiadaných v priamom acyklickom grafe (DAG) riadenom závislosťami
  - definície pracovného toku sú napísané v hPDL(Hadoop process definition language) jazyku, akcie pracovného toku spúšťajú úlohy na servroch, po dokončení systém oznámi jej ukončenie a pokračuje sa nasledujúcou akciou v pracovnom toku


start
M/R job
OK
ERROR
kill
fork
M/R job
Pig job
join
MORE
decision
ENOUGH
M/R job
Java
FS job
end
Control-flow nodes
(start, kill, end | fork, join, decision)
Action nodes
(map reduce, pig, hive, distcp, java, fs, sub-workflow, shell, ssh, email)