HÁZI FELADAT

Házi Feladat

- 3 fős csapatok
  - Javasolt: legyen benne > másodéves informatikus
- Feladatválasztás listából
  - Eseti elbírálással: “hozott” feladat
  - Kiírások: honlap
  - Jelentkezés: form
- Teljesítés
  - Konzulens minden HF-hez; 2 db konzultációt biztosítunk
  - 11. héten előzetes specifikáció és feladatpontosítás leadása
  - 5 perces bemutató-előadás az utolsó héten
  - PDF/HTML dokumentáció és leadása
  - Forráskódok, szkriptek leadása (reprodukálhatóság)

Házi Feladat

- A tipikus házi feladat

- Kaggle adatkészlet, de ***nem*** kaggle feladat

Expedia szállásfoglalás	**http://www.kaggle.com/c/expedia-personalized-sort/data**	**1 GB**	térkép alapú vizualizáció kidolgozása a foglalások cél szerinti elemzésére
			(src,dest) "áramlásokkal" (változó vastagságú nyilak) annotált térkép alapú vizualizáció kidolgozása
			térkép alapú vizualizáció kidolgozása, hogy az egyes országok lakói mennyit költenek (átlagosan és összesen) utazásra
			EDA: gyermekek számának hatása az úthosszra

Házi Feladat

- 3 főre ezek a kreditszámnak megfelelő feladatok

- Javasolt megközelítés
  - “number crunching”: választott technológia
  - Vizualizáció/EDA (as applicable): R

- Néhány kakukktojás (pl. klaszterezés)

Házi Feladat

- Opciók: megközelítés és technológia
  - MapReduce
    - **RHadoop (local backend)**, **Cloudera QuickStart VM + RHadoop**, **Rhadoop on Amazon EMR**, Amazon EMR, IBM BlueMix Hadoop, akármelyik Hadoop-as-a-Service
  - Streaming (if applicable)
    - IBM BlueMix Streaming Analytics, Amazon Kinesis, …
  - Spark
    - itt is van “local backend”
    - unsupported!
  - SQL
    - Eseti jelleggel, pl. HAWQ vagy IBM BlueMix dashDB
- A lényeg: nem kötjük meg a kezeteket
- BlueMix hozzáférés folyamatban

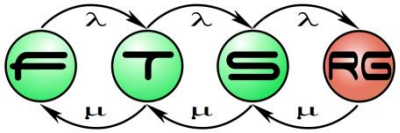
ZH

- 12. héten, az óra időpontjában
- “Ellenőrző kérdések” ezen a héten

Stream Processing

„Big Data” elemzési módszerek

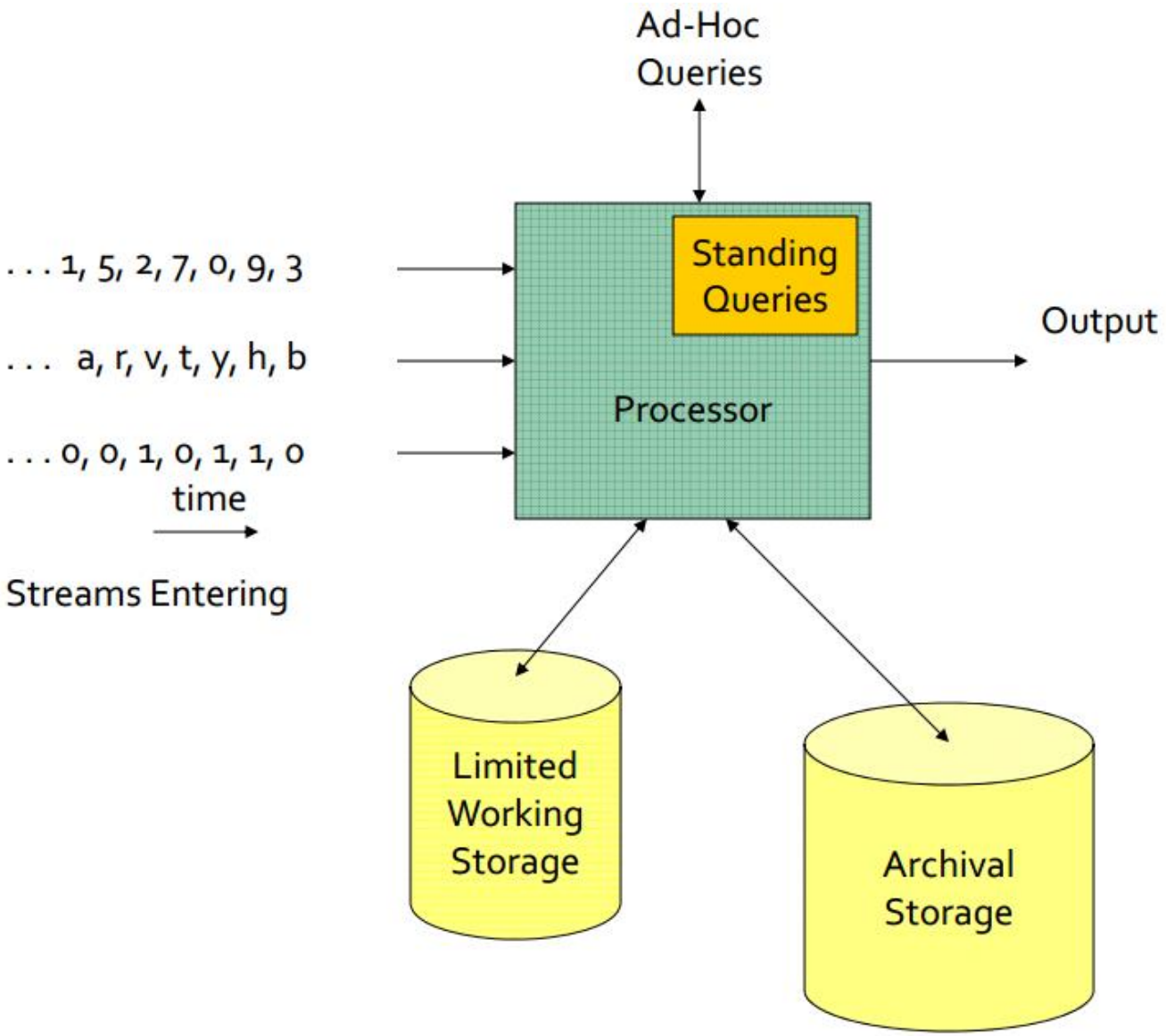
Kocsis Imre
ikocsis@mit.bme.hu
2015.11.04.



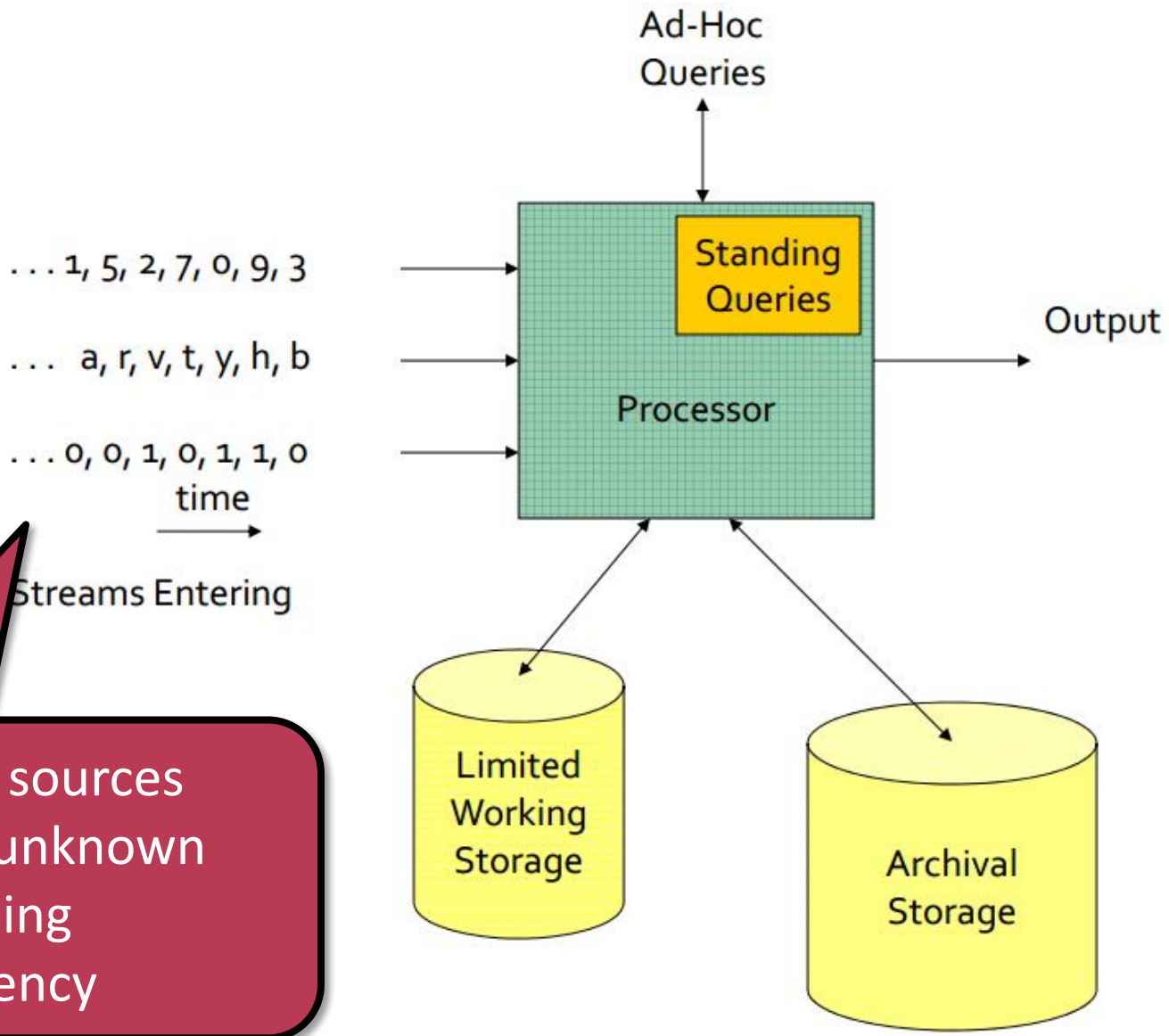
Adatfolyam-források

- Szenzor-adatok
  - 1 millió szenzor x 10/s x 4B
- Képek
  - Szatelitek: n TB/nap
- Internetes szolgáltatások
- Hálózati forgalom
- Tőzsdei adatok
- …

Stream processing (vs „at rest” Big Data)

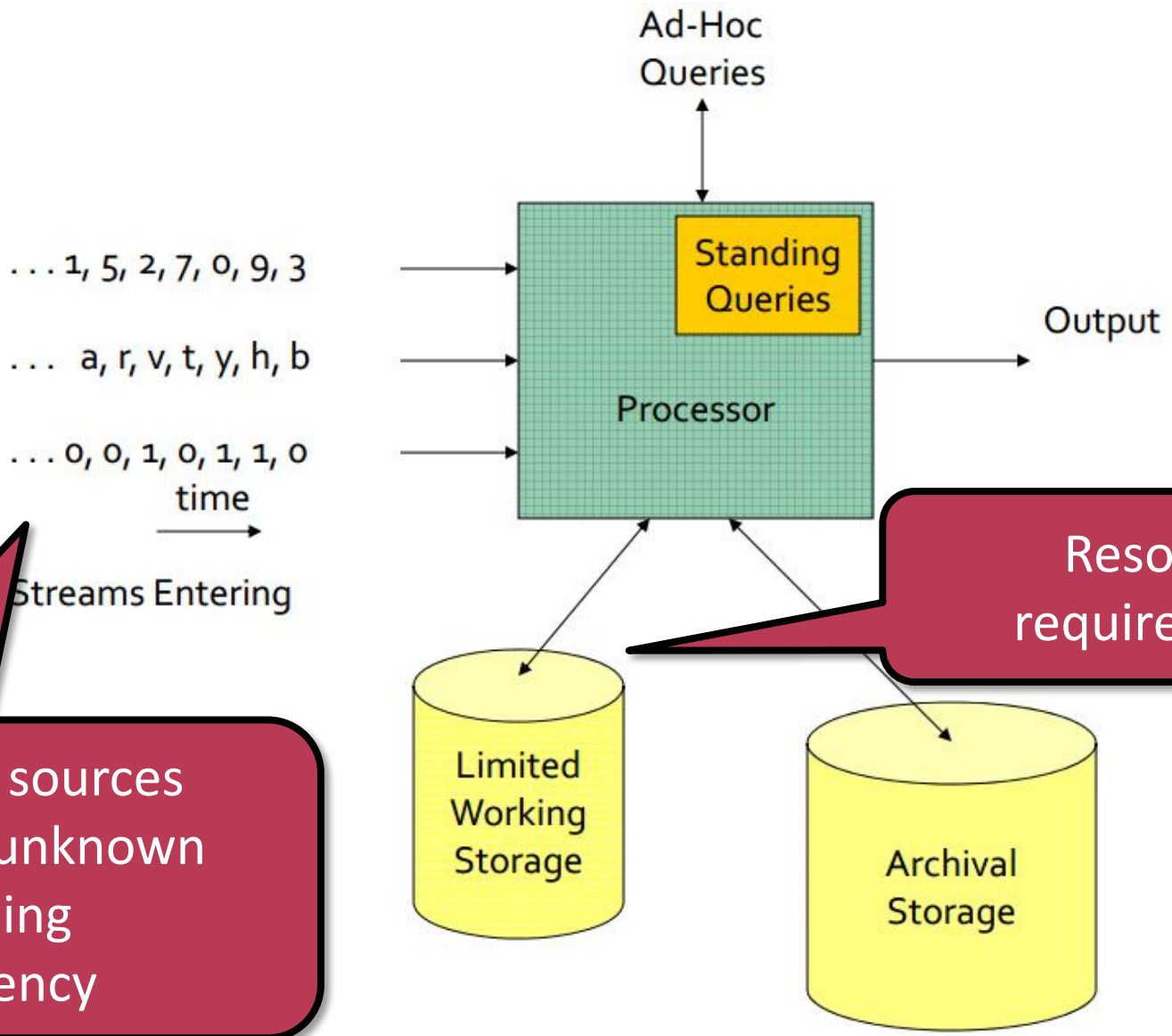


Stream processing



1. Many sources
2. With unknown sampling frequency

Stream processing

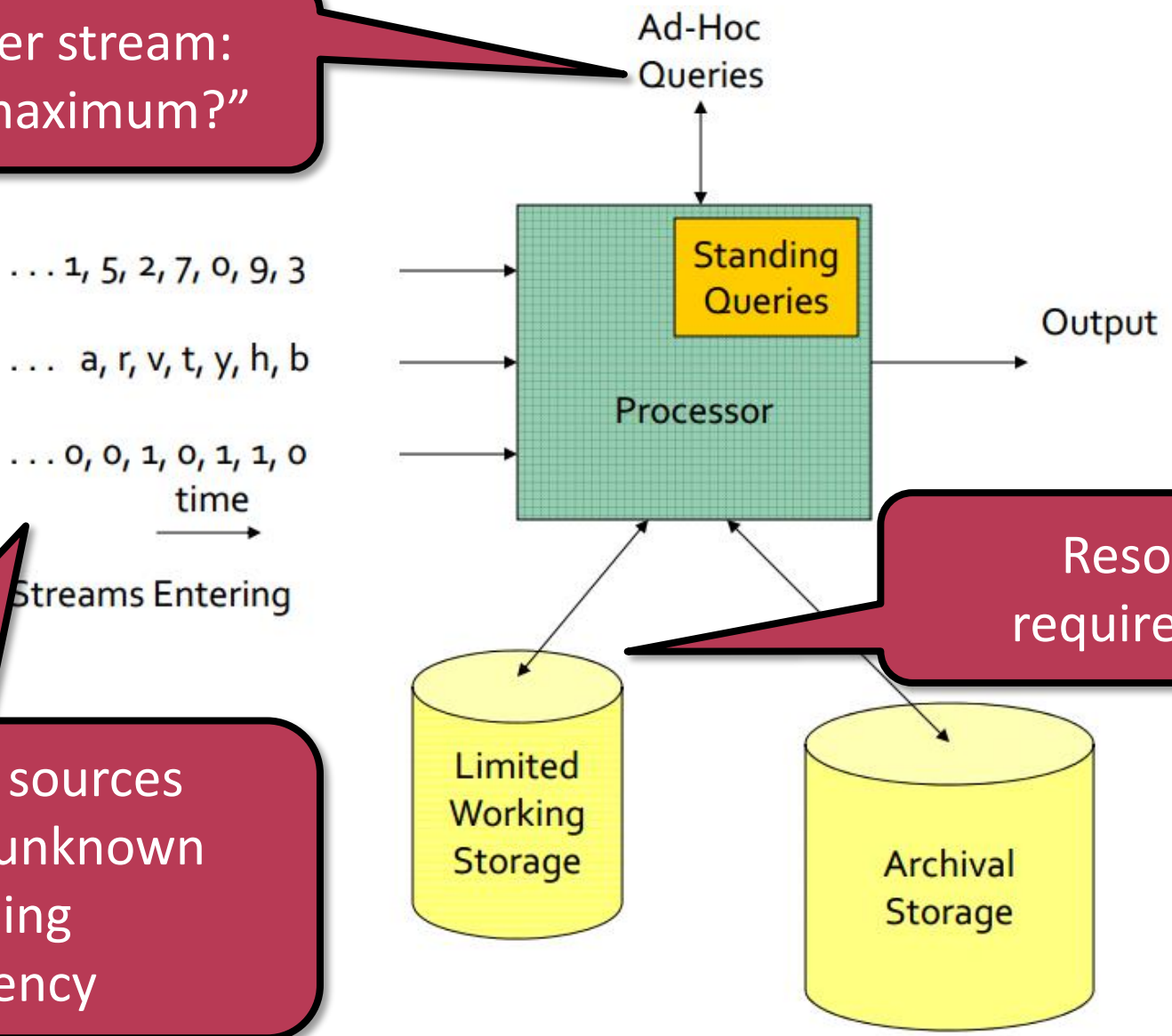


Resource requirements

1. Many sources
2. With unknown sampling frequency

Stream processing

Once per stream:
„Local maximum?”



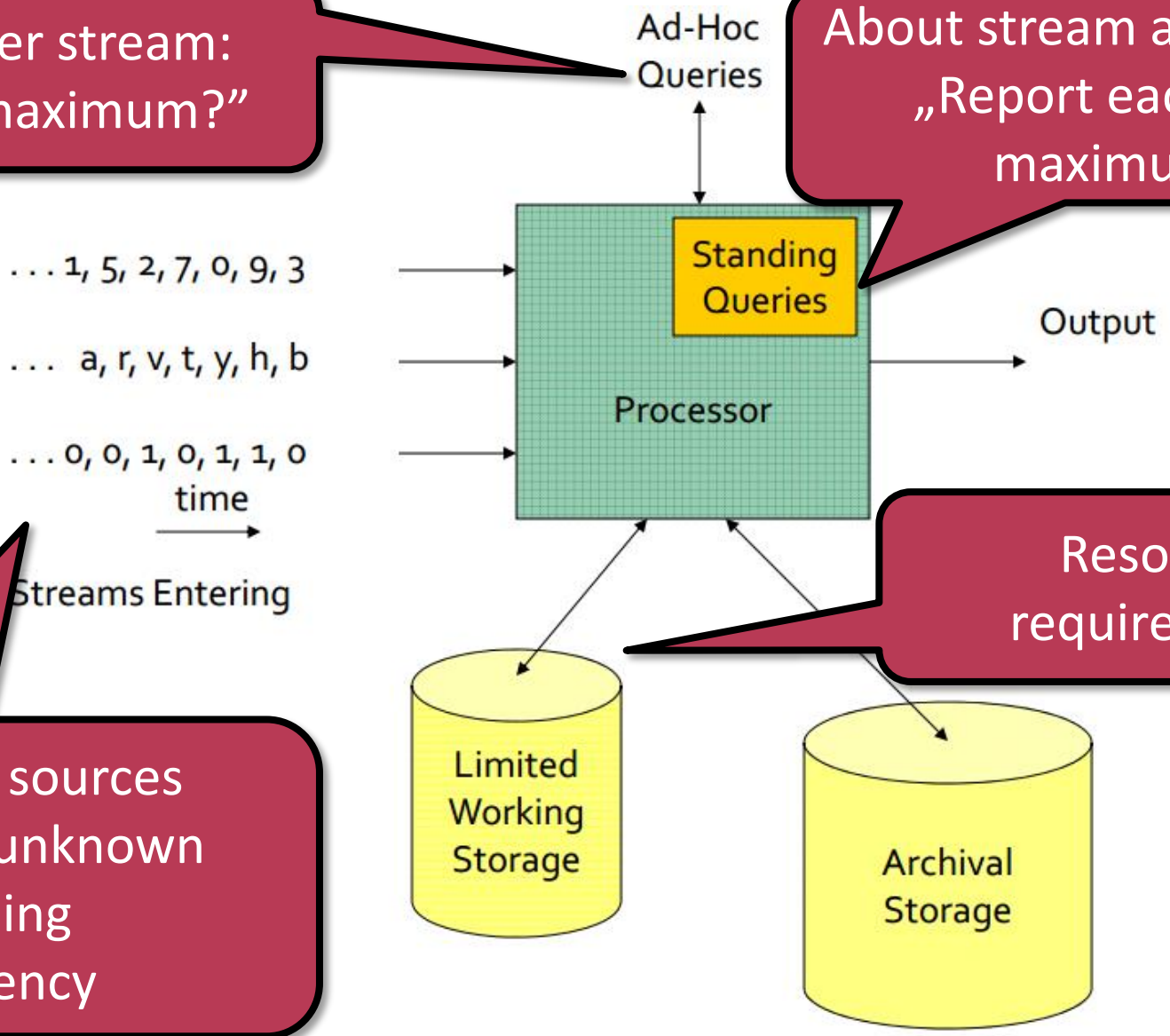
Resource requirements

1. Many sources
2. With unknown sampling frequency

Stream processing

Once per stream: „Local maximum?”

About stream at all times: „Report each new maximum”

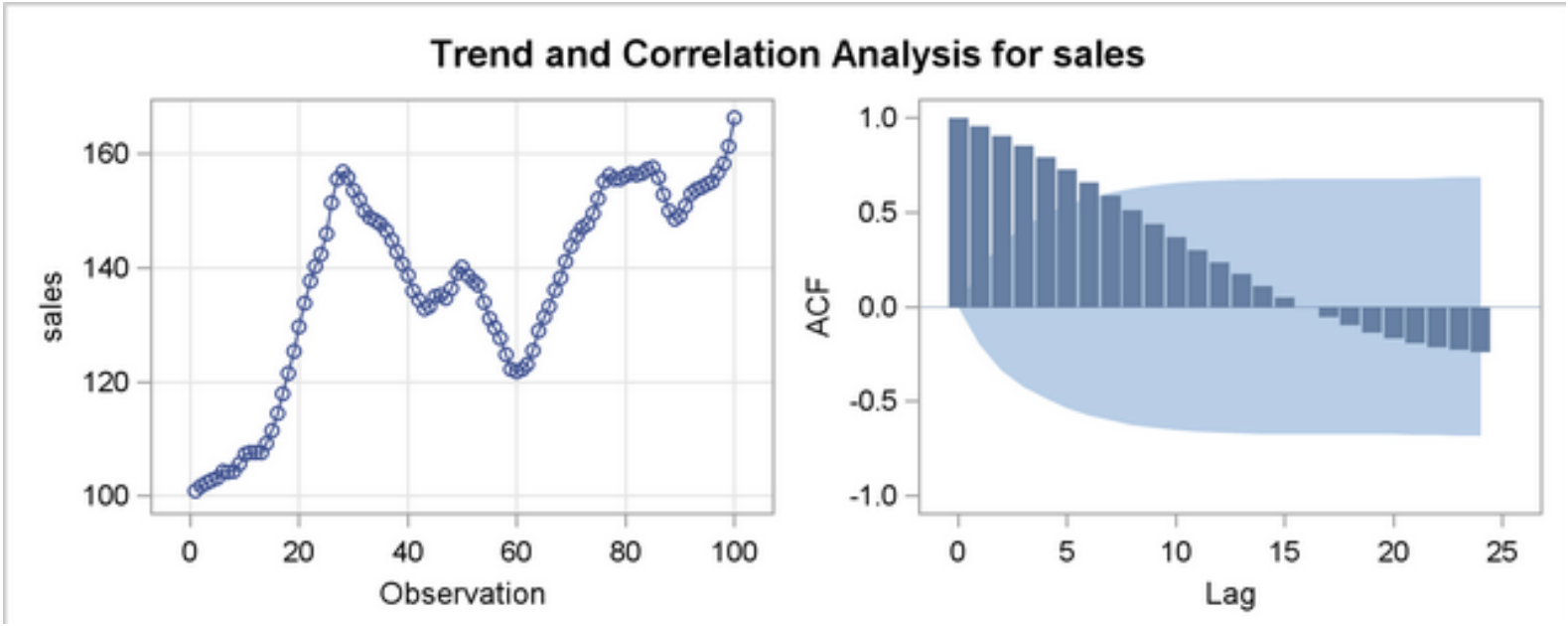


Resource requirements

1. Many sources
2. With unknown sampling frequency

Typically sliding window approches

- Autocorrelation methods
  - Where do we differ from the predicted value?
  - Where does the autocorrelation model change?

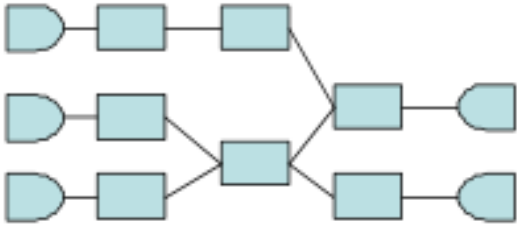


Feldolgozás: időkorlát!

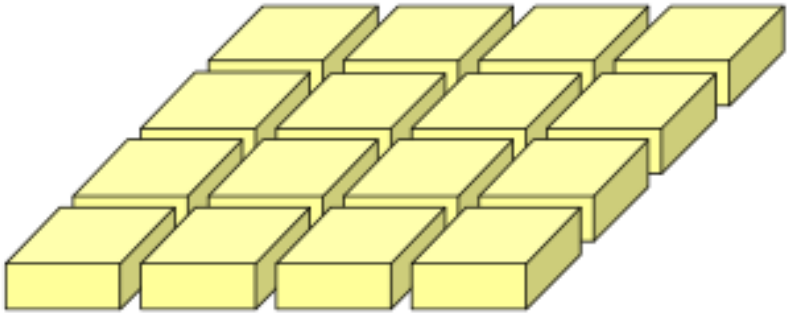
- Diszk nem használható
- Megengedett memóriaigény: korlátos
- Elemenkénti számítási igény: korlátos

- Szokásos megoldások:
  - n-esenkénti (*tuple*) feldolgozási logika
  - Csúszóablakos tárolás és feldolgozás
  - Mintavételezés
  - Közelítő algoritmusok
  - WCET-menedzsment: skálázási logikán keresztül
    - Illetve lehet heurisztika/mintavétel-hangolás is, de az nehéz

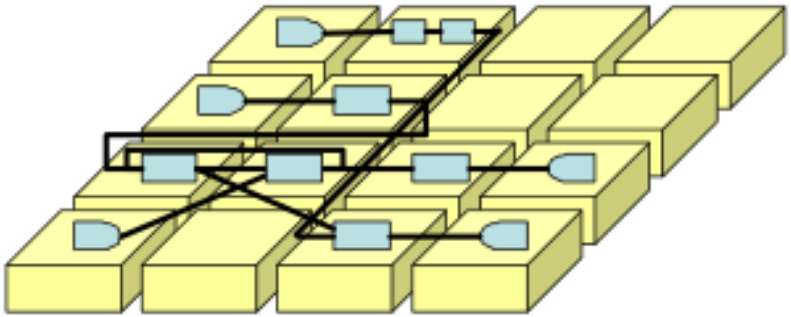
IBM InfoSphere Streams



Logical design:
Application topology



Physical design:
Hardware environment



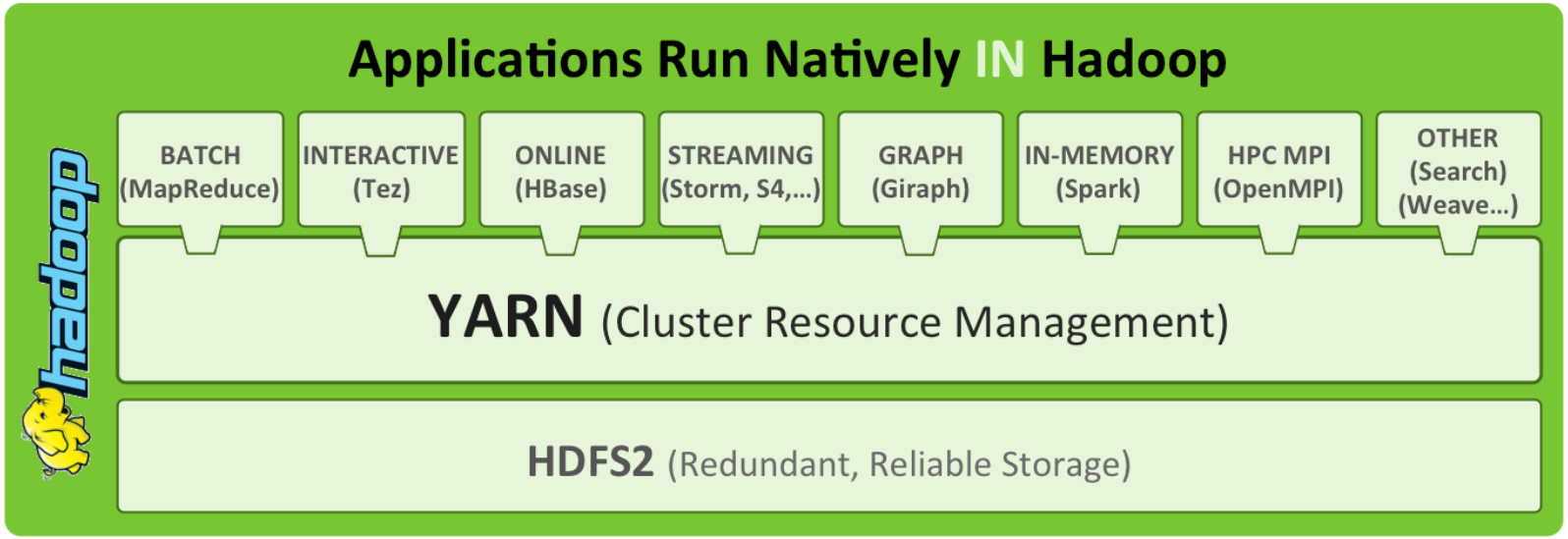
Deployment of application to hardware

Forrás: [2], p 76

Eszközök (néhány!)

- LinkedIn Samza
- Storm

Ábra forrása: [3]

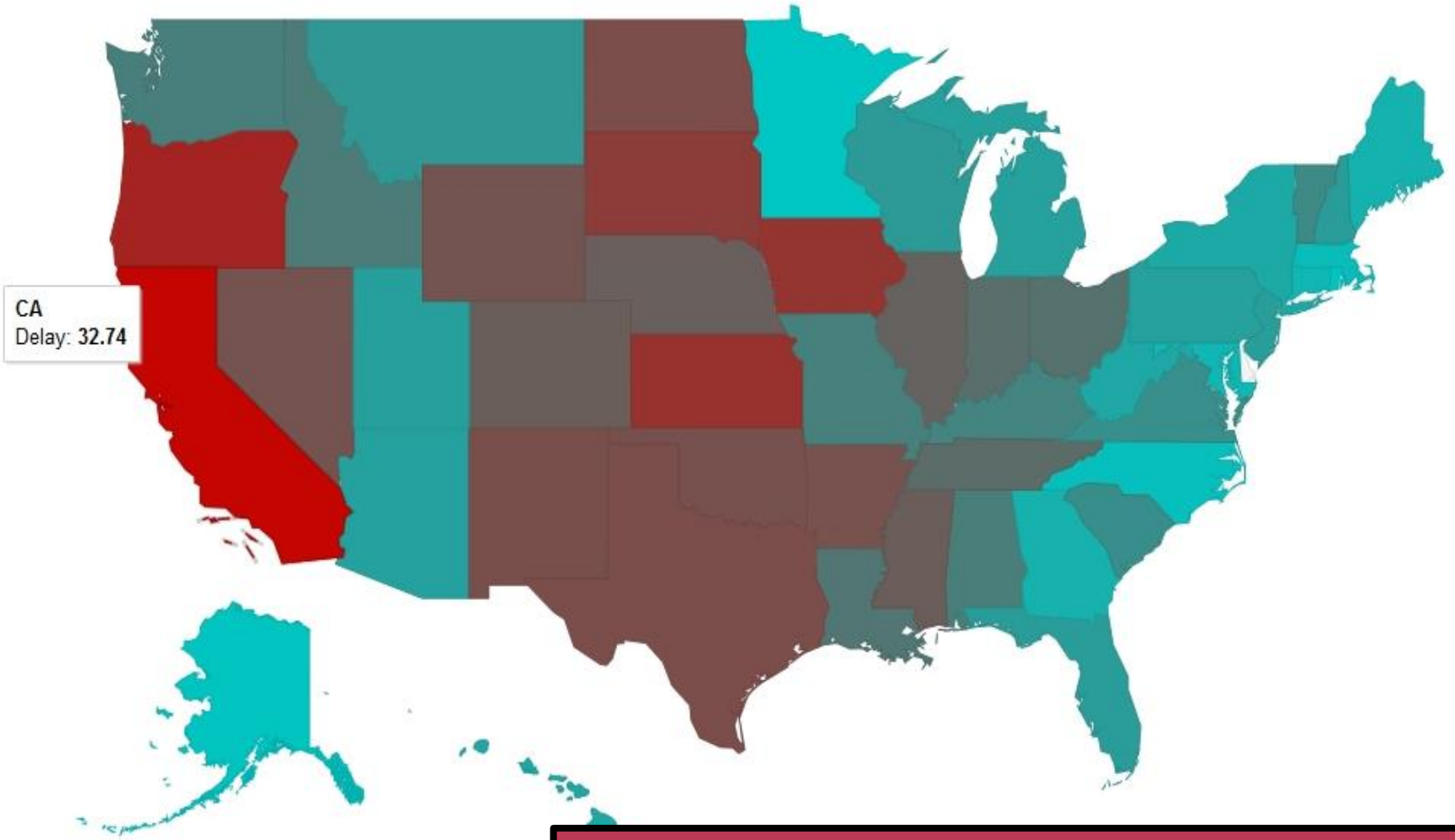


- IBM InfoSphere Streams
- Amazon Kinesis
- …

+ kapcsolódó projektek

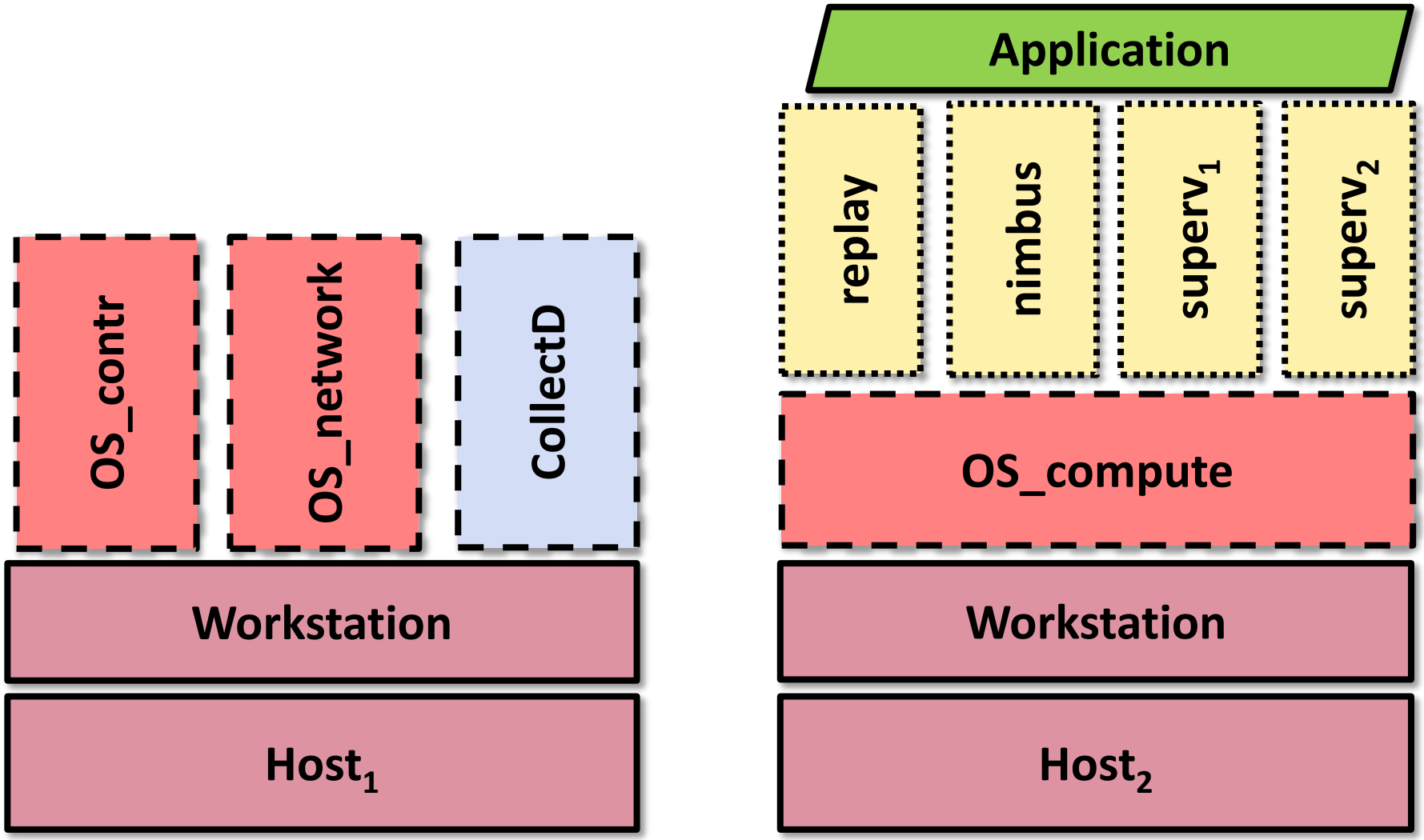
MINTAALKALMAZÁS

Date: **2014-02-06T08:15:00 - 2014-02-10T08:25:00**

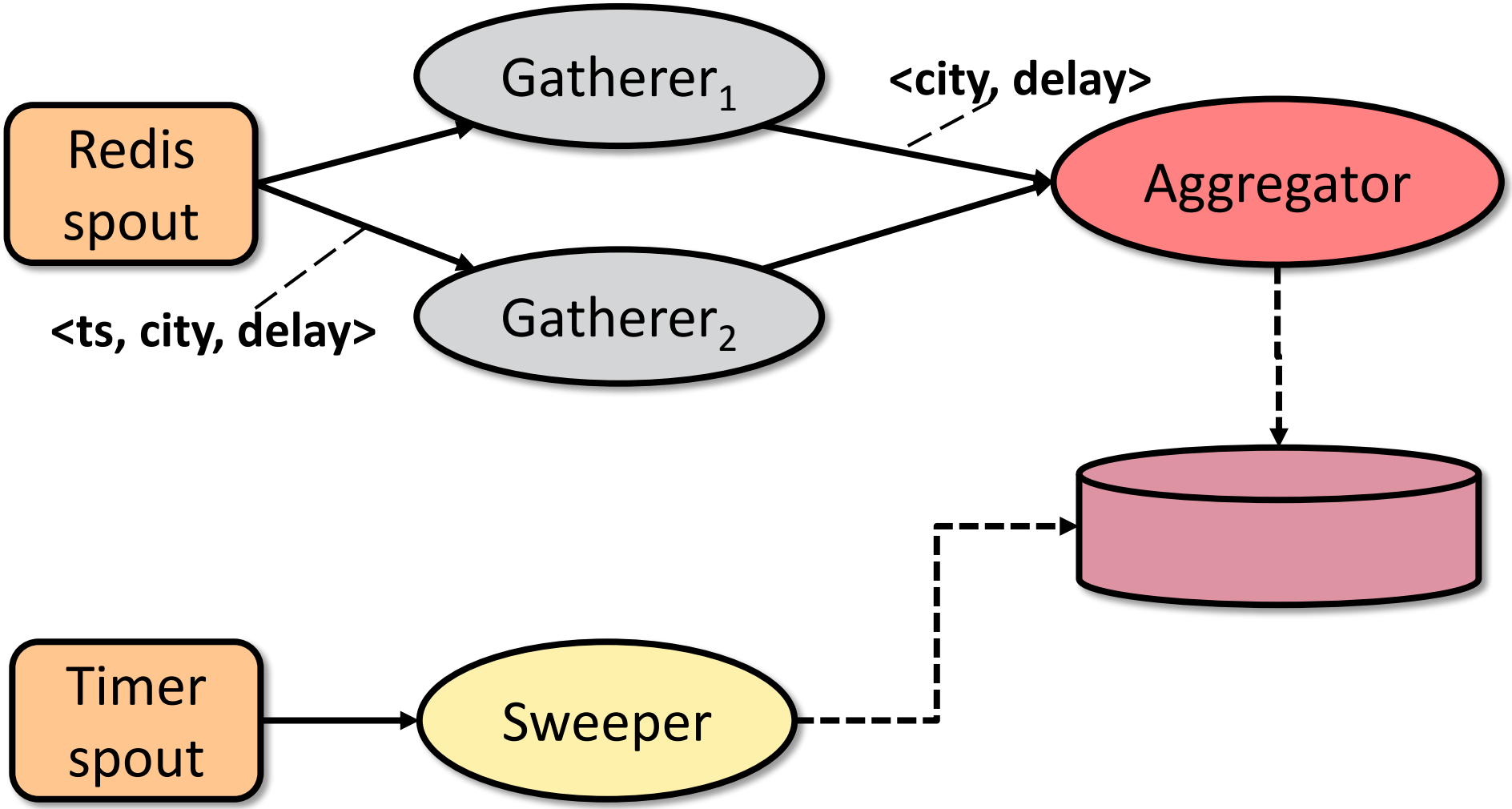


USA polgári légiközlekedés késési adatai

Experimental environment

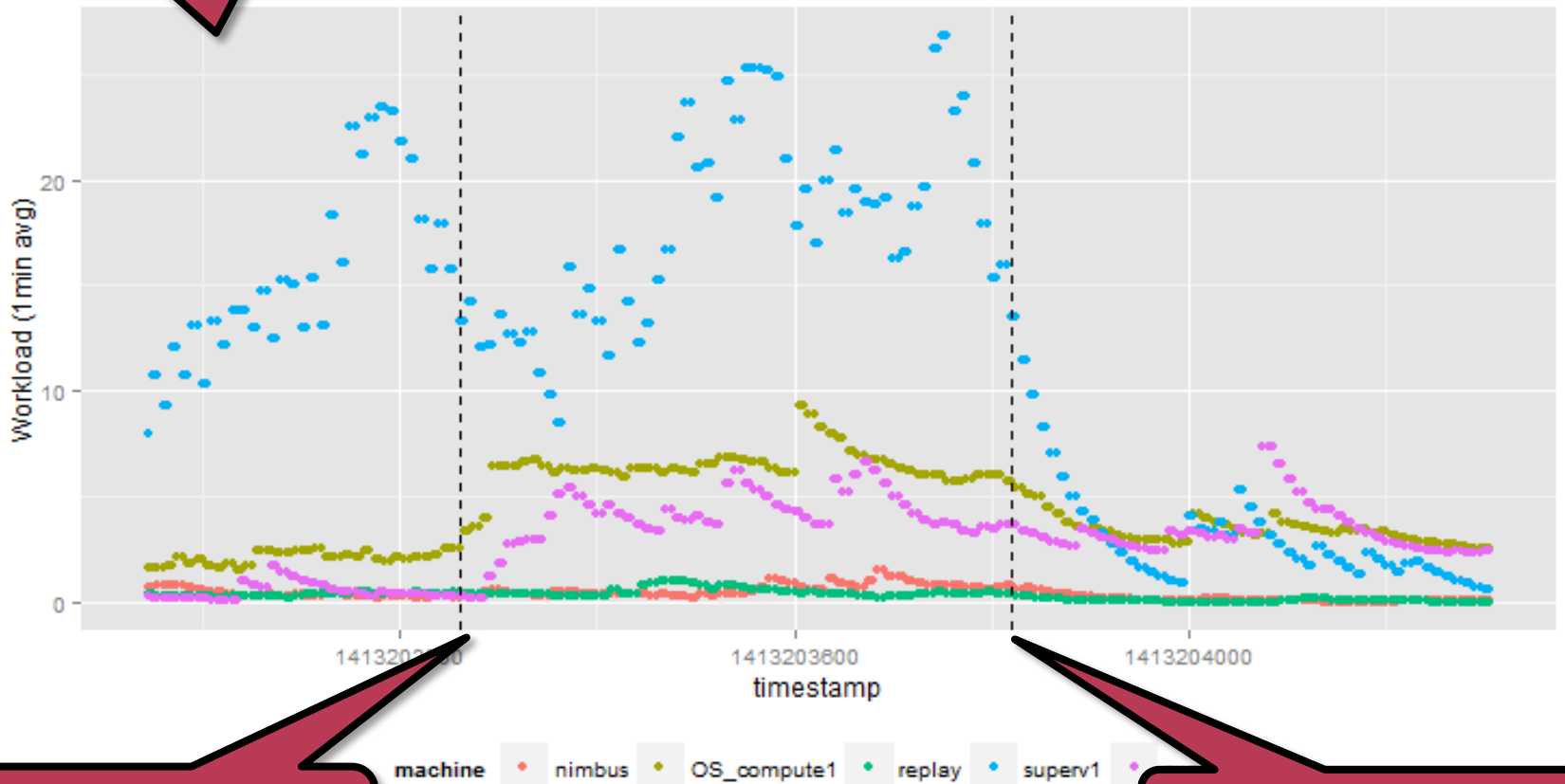


Application topology



Workload

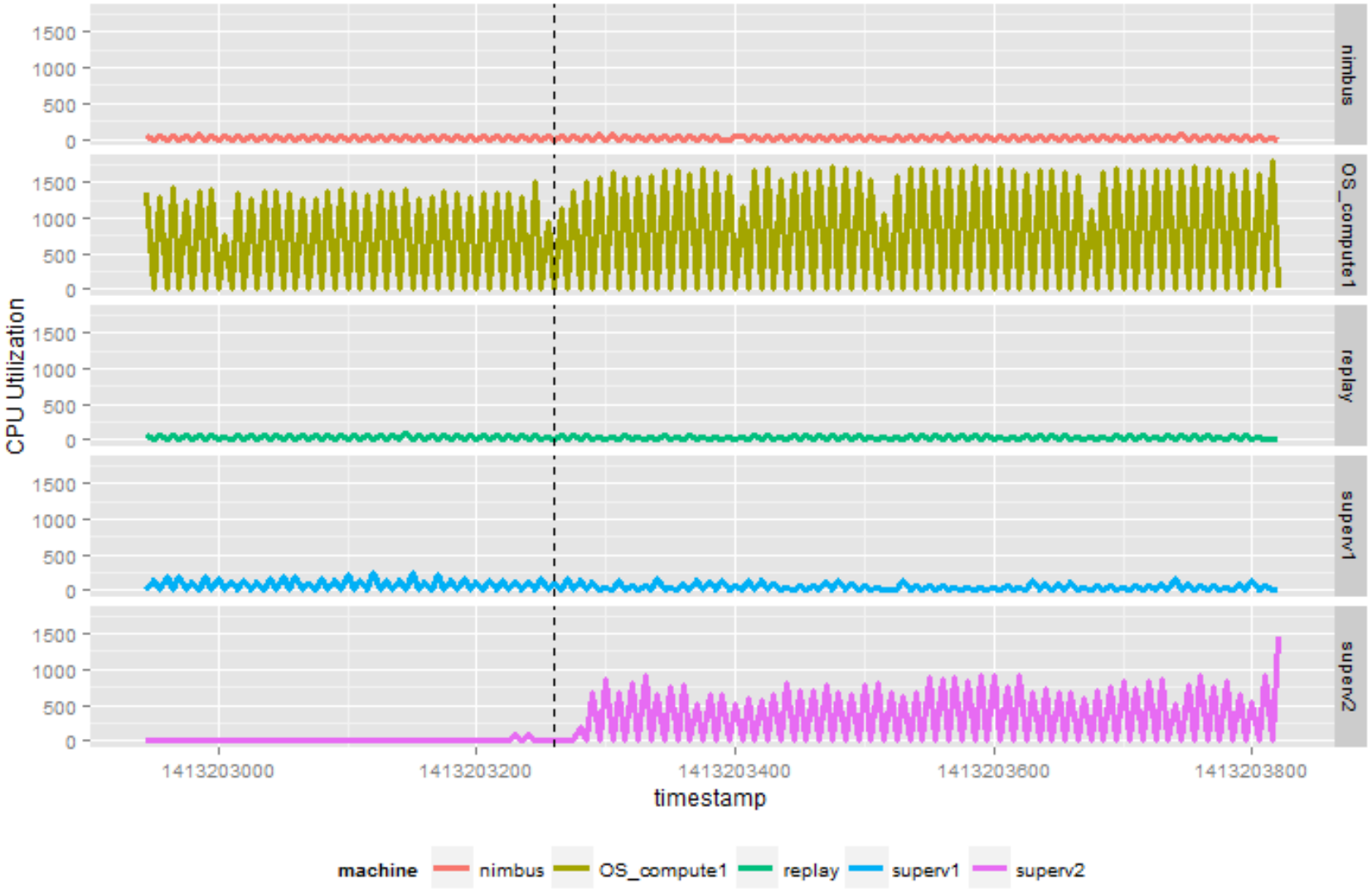
Baseline workload



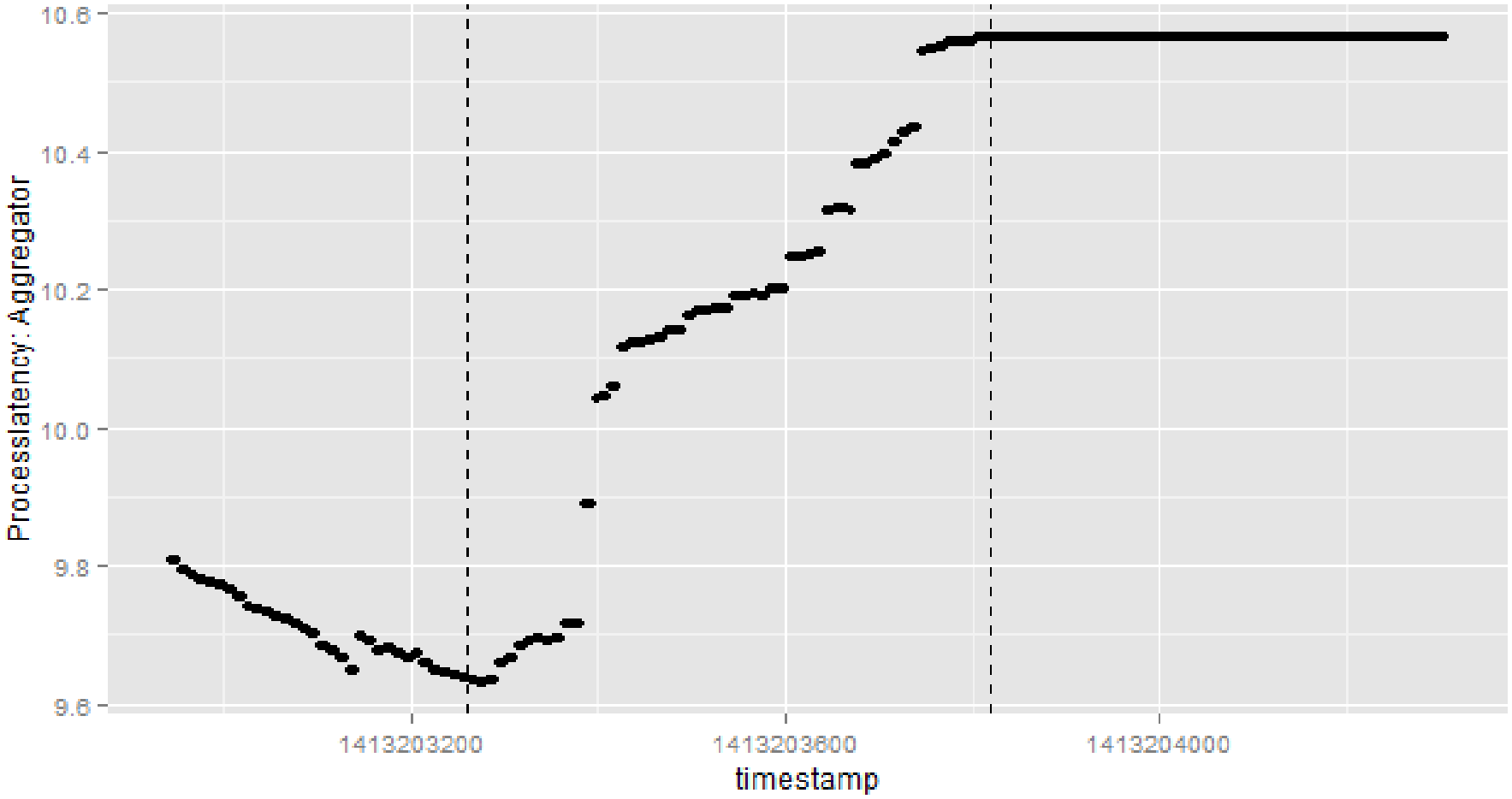
Start of stress

End of stress

CPU utilization

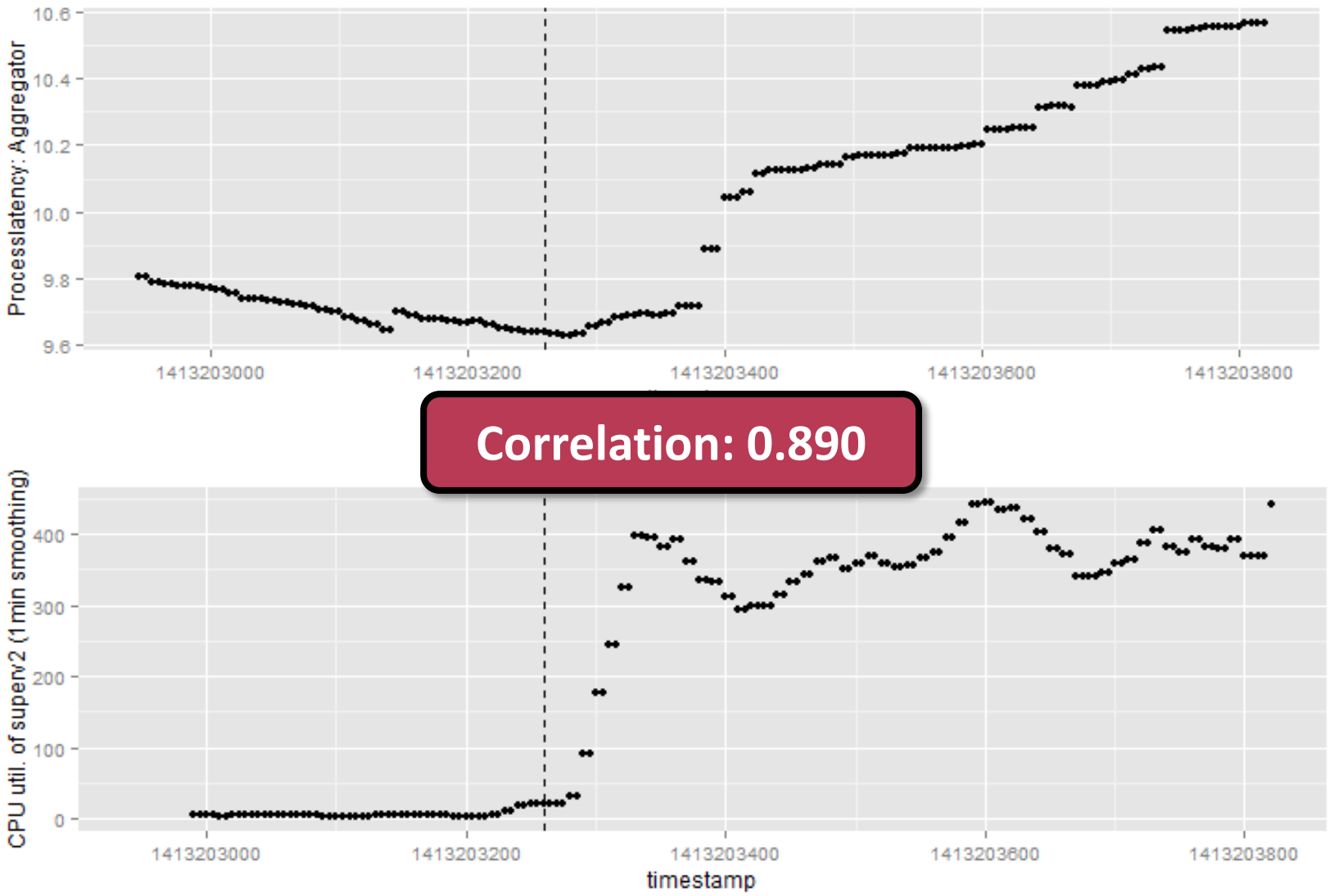


Process latency



Relationship with guest resource usage?

Process latency



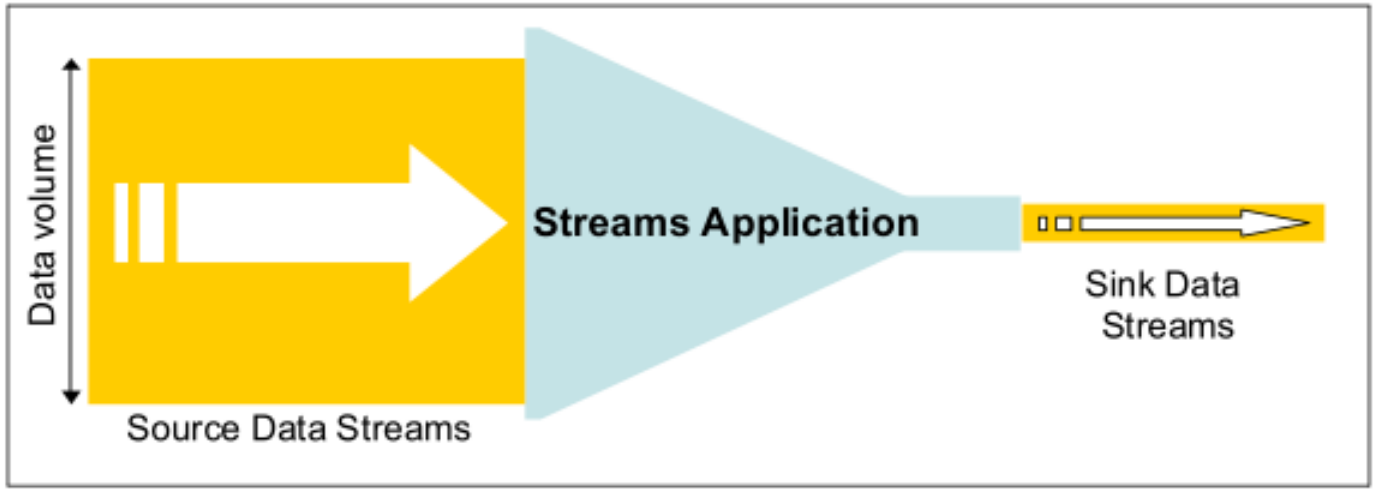
ALKALMAZÁSI MINTÁK

Alkalmazás-osztályok

Class	Sources	Output	Dependency	Performance
Throughput	Known, Fixed	Analysis		Throughput
Latency	Known, Fixed	Opportunities		Latency
Discovery	Partly known, Dynamic	Novel observation	Hypotheses, Stored data	
Prediction	Known, Multiple	Warnings	Human input	Accuracy
Control	Known, Controllable	Controls, Feedback		Latency

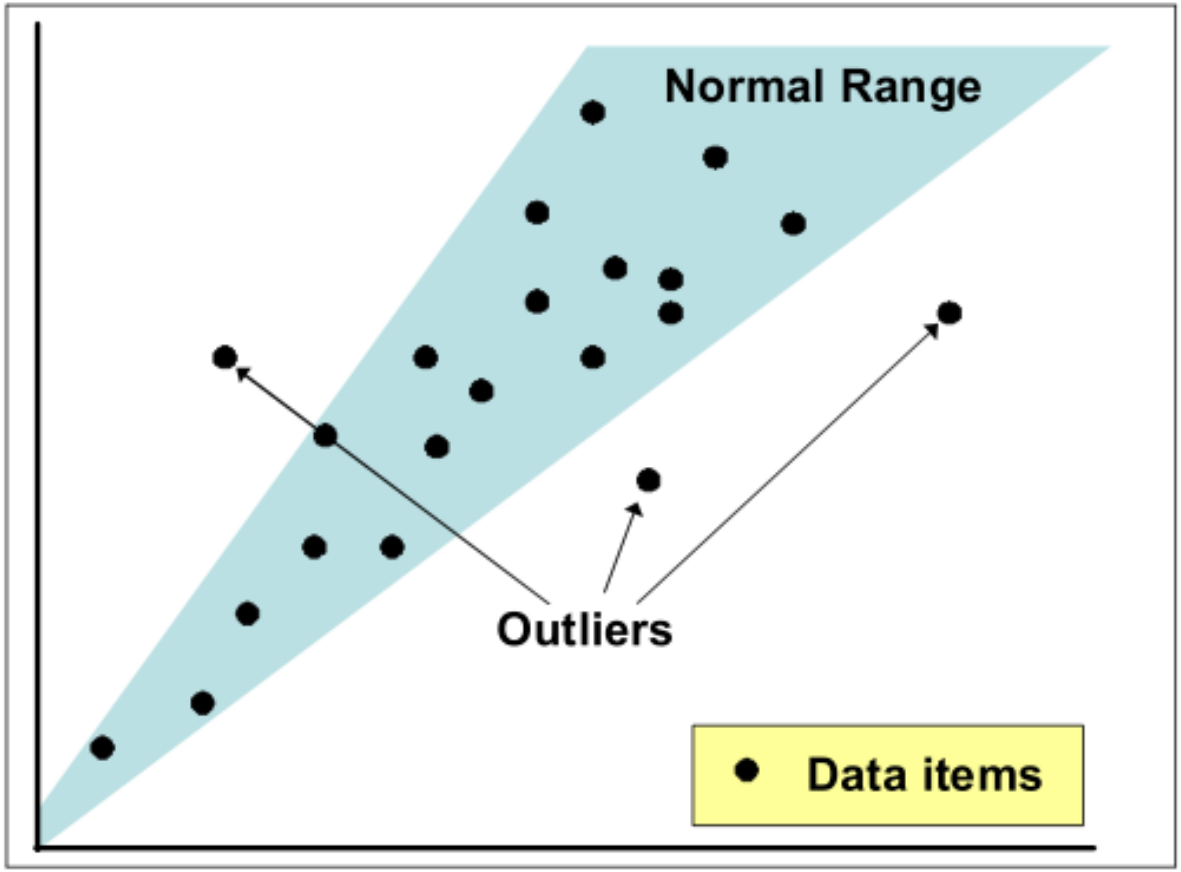
Forrás: [2], p 80

Tervezési minták: filter

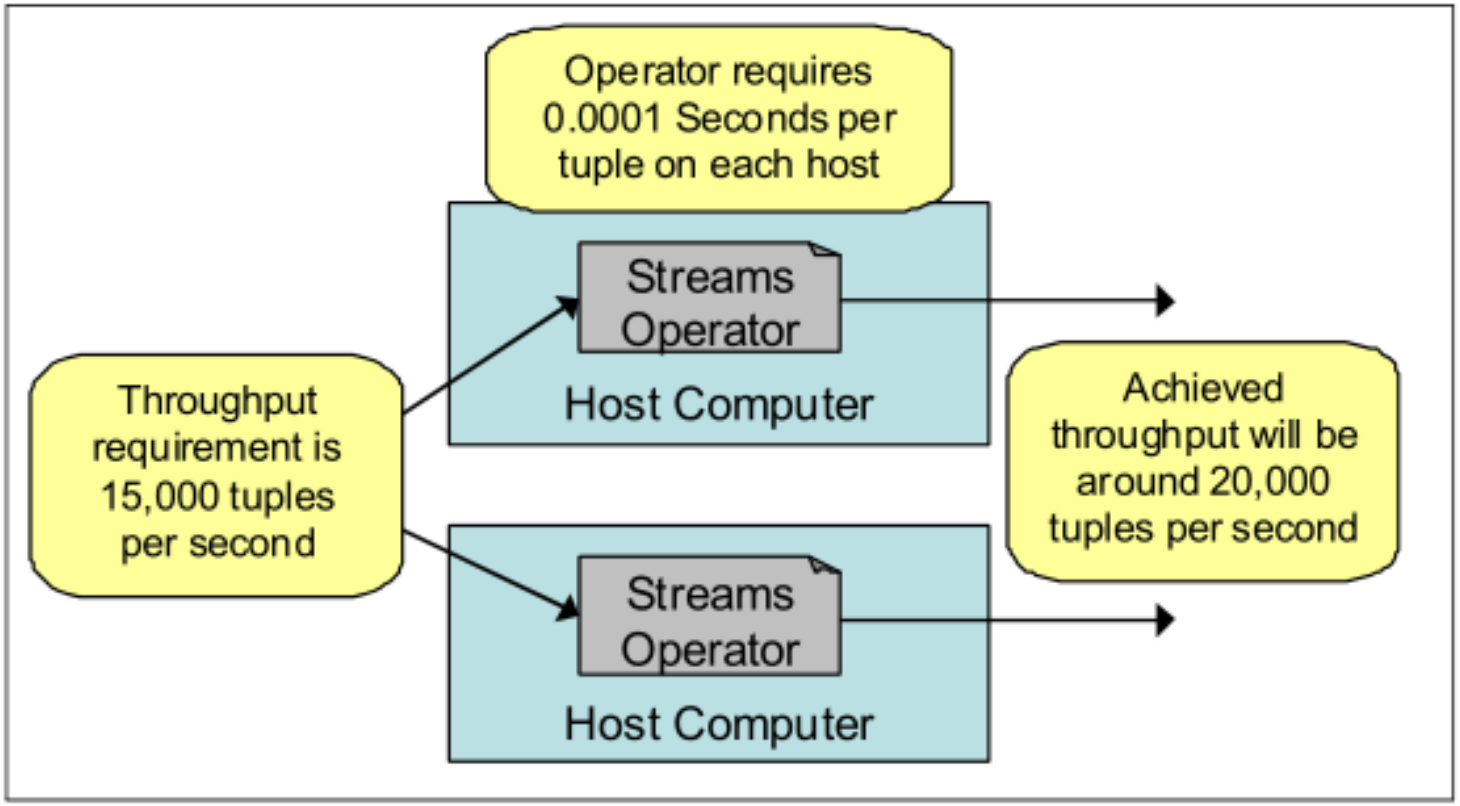


Forrás: [2], 3.2 alfejezet

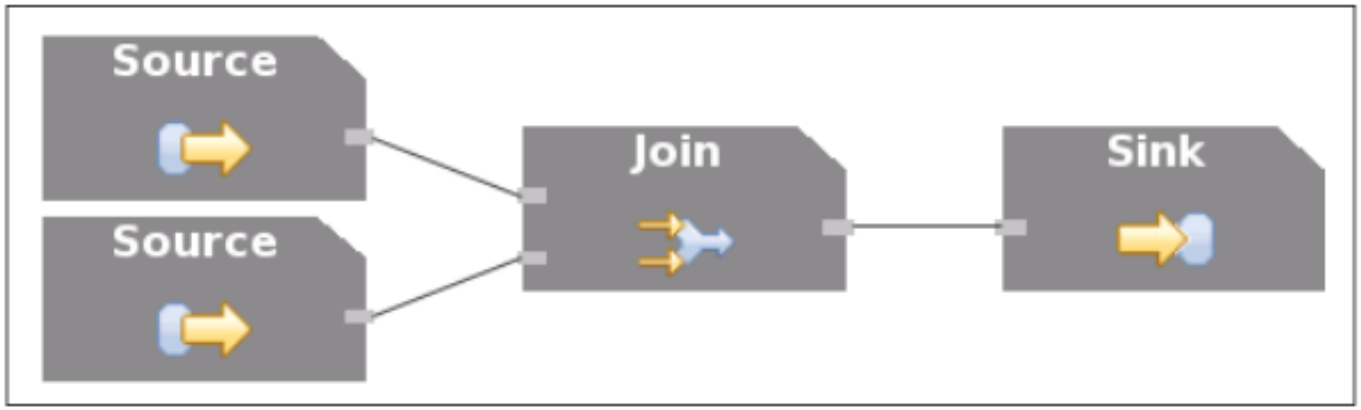
Tervezési minták: outliers



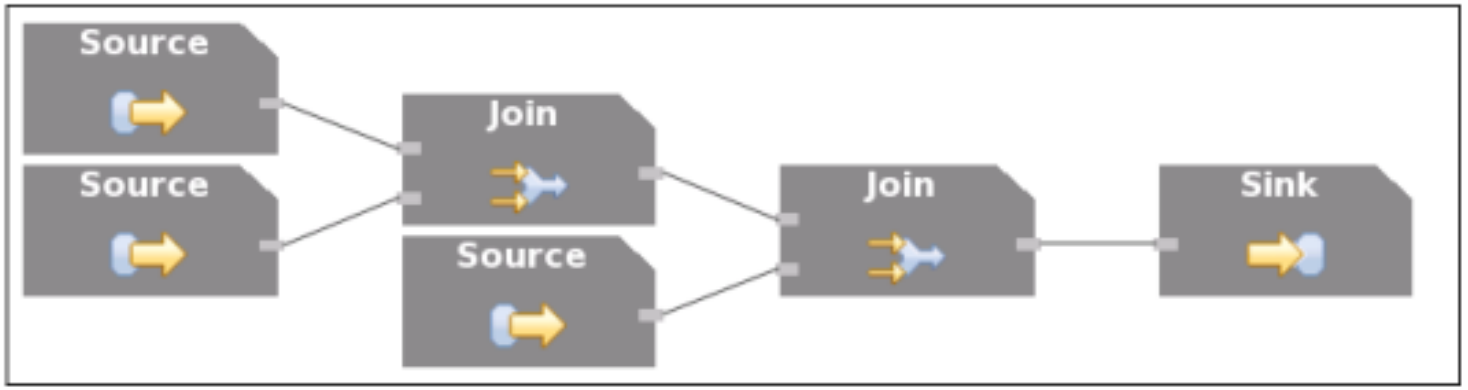
Tervezési minták: parallel



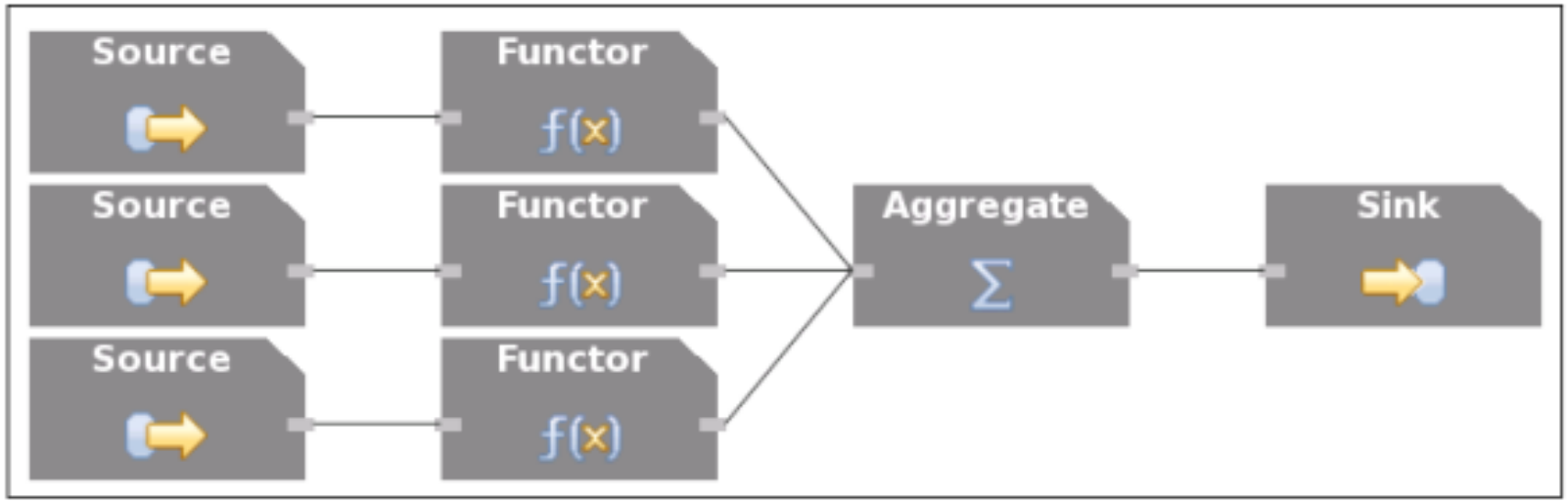
Tervezési minták: supplemental data



Tervezési minták: consolidation



Tervezési minták: merge



R INTEGRÁCIÓ ☺

IBM InfoSphere Streams: R-project Toolkit

- RScript operátor az SPL-ben

```
stream<int32 a, int32 b, int32 c< analyzedStream =
        RScript(inStream) {
        param
                rScriptFileName : "../process.r" ;
                streamAttributes : a, b;
                rObjects : "in1", "in2";
        output
                analyzedStream:
                        c = fromR("out1");
}
```

Forrás: [4]

ALGORITMIKAI SZEMELVÉNYEK

Folyam-algoritmikai szemelvények

- A számítási modellt láttuk
- Fő korlát: adott tár + WCET, „be nem látott” adat

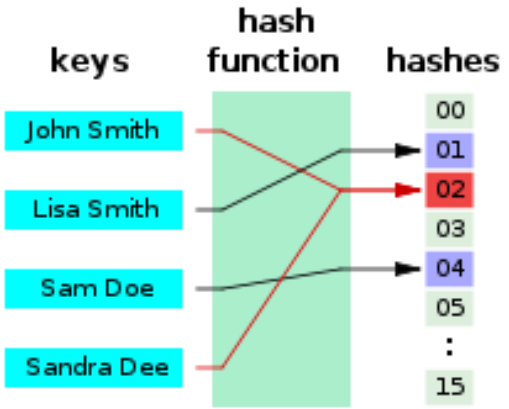
- Néhány tipikus probléma
  - Mintavételezett kulcstér, kulcsok minden értéke
  - „Elég jó” halmazba tartozás-szűrés kicsi leíróval
  - „Count distinct” *korlátos tárral*
  - Momentumok

- Részletes tárgyalás: [1] 4. fejezete

Kitérő: hash-függvények

- Cél: *U* nem rendezett univerzum elemein (átlagosan) gyors keresés, beszúrás, törlés, módosítás

- Eszköz: *h* hash függvény, ami rekordhoz logikai címet rendel
  - A címtartomány jellemzően sokkal kisebb, mint $|U|$
  - Ütközések: $K \neq K' \nRightarrow h(K) \neq h(K')$
  - Vödrös hash-elés, …



http://en.wikipedia.org/wiki/Hash_function

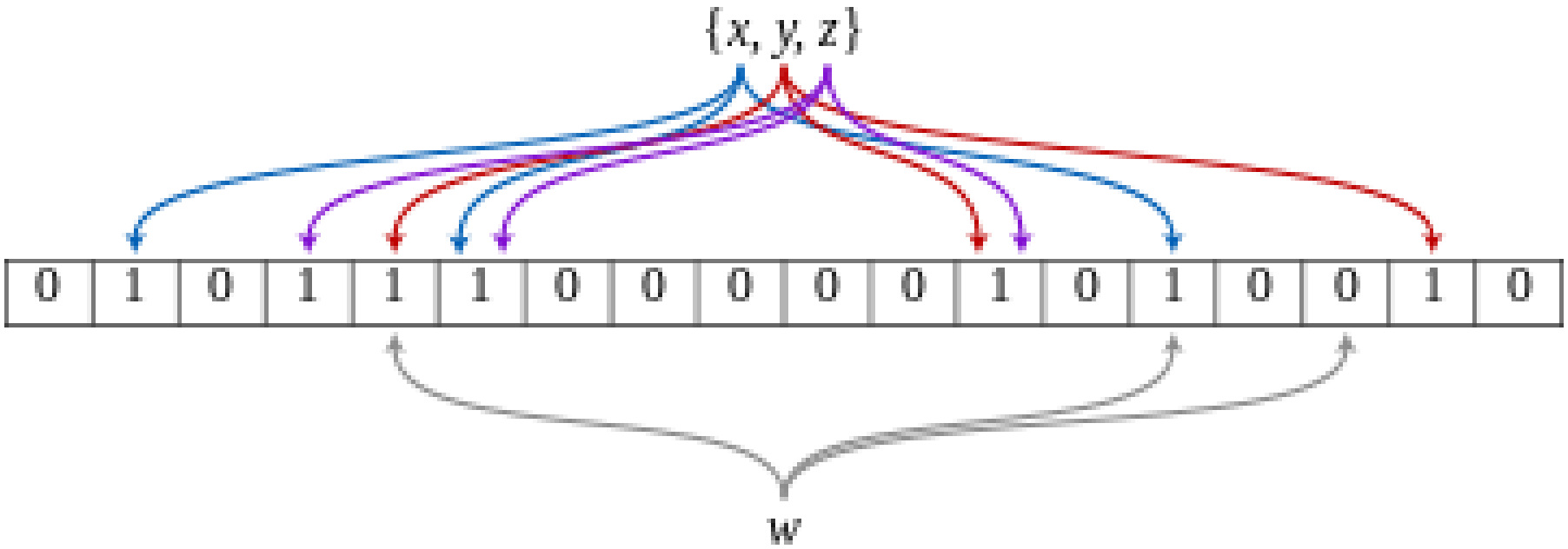
Hash-függvények: jellemző követelmények

- Alkalmazási területenként eltérőek!
  - Kriptográfia ←→ indexelés adattároláshoz
- Néhány tipikus követelmény
  - Determinizmus
  - Uniformitás
  - Meghatározott értékkészlet
  - Folytonosság
  - Irreverzibilitás („egyirányú” függvény)
- http://en.wikipedia.org/wiki/List_of_hash_functions

Mintavételezés

- Modell:
  - *n* komponensű elemek
  - ezek egy része *key* (pl. **user**,query,time)
  - a kulcsok felett mintavételezünk
- Probléma
  - Egy kulcsnak vagy minden értéke megjelenjen, vagy egy sem
- Megoldás
  - *a/b* méretű mintához *a* (kulcstér)méretű folyamon a kulcsot *b* vödörbe hasheljük
  - A hash-függvény valójában „konzisztens random-generátor”: a < b esetén tárolunk
  - Nem véges minta – kisebb módosítás
- Példa: „a felhasználók mekkora része ismétel meg lekérdezéseket” a felhasználók 1/10 mintáján

Bloom filterek



http://en.wikipedia.org/wiki/Bloom_filter

Szűrés: Bloom filterek

- Bloom filter:
  - *n* bites vektor, kezdetben azonosan 0
  - Hash függvények kollekciója: $\mathrm{h}_1, h_2, \ldots h_k$. Mindegyik kulcsokat rendel *n* vödörhöz (a vektor elemeinek felelnek meg).
  - S: kulcshalmaz ($|S| = m$)

- Cél: minden $K \in S$ átengedése, a ***legtöbb*** $K \notin S$ kiszűrése – tárhely-hatékonyan

- Példa: spam email-cím alapján

Szűrés: Bloom filterek

- Indulás: minden j bit-et 1-re állítunk, amire van h_i és $K \in S,$ hogy $h_i(K) = j$

- Kulcs tesztelése: minden függvény eredménye 1 értékű bitbe visz-e
  - Igen: továbbengedés (lehet hogy S-ben)
  - Nem: dobás (nem lehet S-ben)

- Kaszkádolható!

- False positive valószínűség: lásd könyv (darts-modell)

Bloom filterek: néhány tétel

- Hibás pozitív valószínűség (uniform hashekkel):
  - $\approx (1 - e^{-km/n})^k$

- Optimális hashfüggvény-szám
  - $k = \frac{n}{m} ln2$

„Count-Distinct”: a Flajolet-Martin algoritmus

- N.B. nem-algoritmikai megoldások is működhetnek
- Legyen egy „bit-sztring” hash-függvénynek több kimenete, mint az univerzum elemei
  - Pl. 64 bit elég az URL-ekhez
- Ezekből “sokat” alkalmazunk
- $h(a)$ a folyam-elemre r 0-ban végződik (*tail length*). Legyen ezek maximuma R.
- Count-Distinct közelítés: 2^R
  - Ha $m \gg 2^r$, akkor szinte biztos van legalább r hosszú farok
  - Ha $m \ll 2^r$, akkor szinte biztos nincs legalább r hosszú farok
- Sok hash függvény, kis csoportok (legalább $c \log_2 m$) átlaga, ezek mediánja

„Count-Distinct”: a Flajolet-Martin algoritmus

- Intuitívan:
  - Egy a –ra $h(a)$ legalább r 0-ban végződik: 2^{-r} val. (Uniform hash azért nem árt.)
  - m különböző elem, egyikükre sem legalább r hosszú a tail:
$$(1-2^{-r})^m$$
  - Átírható: $\left((1-2^{-r})^{2^r}\right)^{m2^{-r}}$
  - Elég nagy r-re a belső tag $\approx \frac{1}{e}$
  - Nincs elem legalább r hosszú tail-el: $\mathrm{e}^{-m2^{-r}}$ val.
  - m jóval nagyobb/kisebb mint 2^r: “biztos van hosszabb”, “biztos nincs kisebb”
  - 2^R nem valószínű, hogy túl pontatlan lenne

Momentumok

- Rendezett univerzum
- m_i: i-ik elem előfordulási száma
- Stream k-adrendű momentuma (k-ik momentum): $\sum_i (m_i)^k$
- Néhány momentum
  - 0: „count distinct”
  - 1: stream hossza
  - 2: előfordulások négyzetösszege – *surprise number*: eloszlás egyenetlensége
    - V.ö. $I(\omega_n) = -\log(P(\omega_n))$

Az Alon-Matias-Szegedy algoritmus

- Legyen a stream n hosszú,
- Nem tudunk minden m_i-t tárolni,
- Második momentum közelítése,
- Korlátos tárhellyel (több → jobb közelítés)

- Minden X változónkhoz tároljuk:
  - Az univerzum egy elemét: $X.element$
  - Egy $X.value$ egészet.

- Inicializálás: uniform, véletlenszerű választással 1 és n között kisorsolt pozíció elemére

Az Alon-Matias-Szegedy algoritmus

- Minden X-ből lehet becsülni:

$$\mathrm{n} \times (2 \times X.value - 1)$$

- Legyen $e(i)$ a stream i-ik eleme; legyen $c(i)$ ezen elem előfordulási száma az i-ik pozíciótól

$$E(n(2 \times X.value - 1)) =$$

$$\frac{1}{n}\sum_{i=1}^{n} n \times (2 \times c(i) - 1) =$$

$$\sum_{i=1}^{n} (2c(i) - 1)$$

Az Alon-Matias-Szegedy algoritmus

- A szumma átrendezése az elemekre:

$$\sum_{i=1}^{n}(2c(i)-1) = \sum_a 1+3+5+\cdots+2(m_a-1)$$

- Indukcióval: $1+3+5+\cdots+(2m_a-1) = (m_a)^2$

- Így:

$$E(2\times X.value+1) = \sum_a (m_a)^2$$

Az Alon-Matias-Szegedy algoritmus

- k-ik momentumra:

$$v = X.value \rightarrow n \times (v^k - (v-1)^k)$$

- Kiterjesztés nem véges stream-ekre:
  - Mindig s változót tárolunk, inicializáció
  - Minden új elemet $\frac{s}{n+1}$ valószínűséggel választunk változónak
  - Ha választjuk, egy régit eldobunk

Hivatkozások

- [1] Rajaraman, A., & Ullman, J. D. (2011). Mining of Massive Datasets. Cambridge: Cambridge University Press. doi:10.1017/CBO9781139058452
- [2] International Technical Support Organization. IBM InfoSphere Streams: Harnessing Data in Motion. September 2010. http://www.redbooks.ibm.com/abstracts/sg247865.html
- [3] http://hortonworks.com/blog/hdp-2-0-community-preview-and-launch-of-hortonworks-certification-program-for-apache-hadoop-yarn/
- [4] http://www.ibm.com/developerworks/library/bd-streamsrtoolkit/