Cenni di Piattaforme di Supporto Clustered per Applicazioni Big Data

Università di Bologna
CdS Laurea Magistrale in Ingegneria Informatica
I Ciclo - A.A. 2013/2014

Corso di Sistemi Distribuiti M

Scenari di Applicazioni Big Data e Sfide Tecnologiche Aperte per loro Supporto

Docente: Paolo Bellavista
paolo.bellavista@unibo.it

http://lia.deis.unibo.it/Courses/sd1314-info/
http://lia.deis.unibo.it/Staff/PaoloBellavista/

Data & Data & Data & Management

Centralità dei dati e della gestione di dati

Molti sistemi sono sempre più costituiti e caratterizzati da ***enormi moli di dati da gestire,*** originati da ***sorgenti altamente eterogenee*** e con ***formati altamente differenziati***, oltre a ***qualità estremamente eterogenea (qualità dei dati)***

Esempi tipici:

- Sistemi distribuiti a larga scala
- Sistemi mobili
- Sistemi embedded e Internet of things
- Sistemi cloud

Tipiche aree applicative:

- Scenari ***smart city***
- Applicazioni ***social***
- Large-scale ***streaming*** information
- Data center e ***high-performance computing***
- ***Sistemi e servizi mobili***

Big Data: anche le Aspettative sono Big…

Mercato e investimenti in tematiche Big Data

- **6.3 billion** of USD **2012**
- **48.3 billion** of USD **2018**

atteso ***incremento di 45% per anno***
sia da parte di investimenti pubblici che privati

Overall ICT industry market in 2020

- **5 trillion** of USD **2020**

Guidato e spinto da piattaforme per Mobile broadband, Social business, servizi Cloud e Big Data management&analytics

Anche grande sforzo EU

Diverse iniziative nel framework Horizon 2020, anche connesse alle tematiche **Open** and **Linked data** (soprattutto dati PA, …)

Forse Private Public Partnership (PPP) emergente nei prossimi mesi…

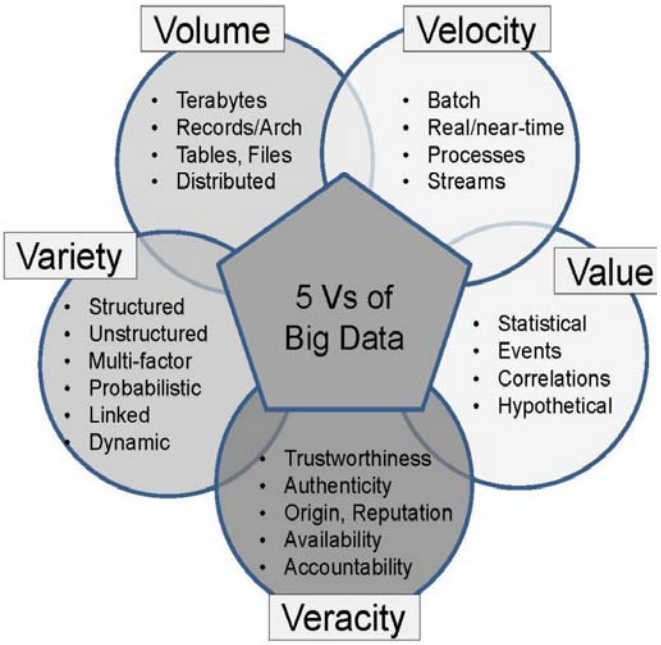
All-Data Vision (non solo Big Data)

Sistemi informativi richiedono una ***visione quality-aware che possa organizzare e rendere efficiente l'intero data lifecycle***

5 V per il nuovo processamento e trattamento dei dati

- **Volume of Data**
- **Variety of Data**
- **Velocity**
- **Value**
- **Veracity**

6 V includendo anche
- **Variability**



All-Data Service and Management

Sistemi informativi moderni richiedono ***gestione quality-aware e processing workflow ottimizzato*** per produrre ***nuovi servizi e incrementare data value***

Visione di base unificante: ***full set di servizi per arricchire e accompagnare l'intero data lifecycle***

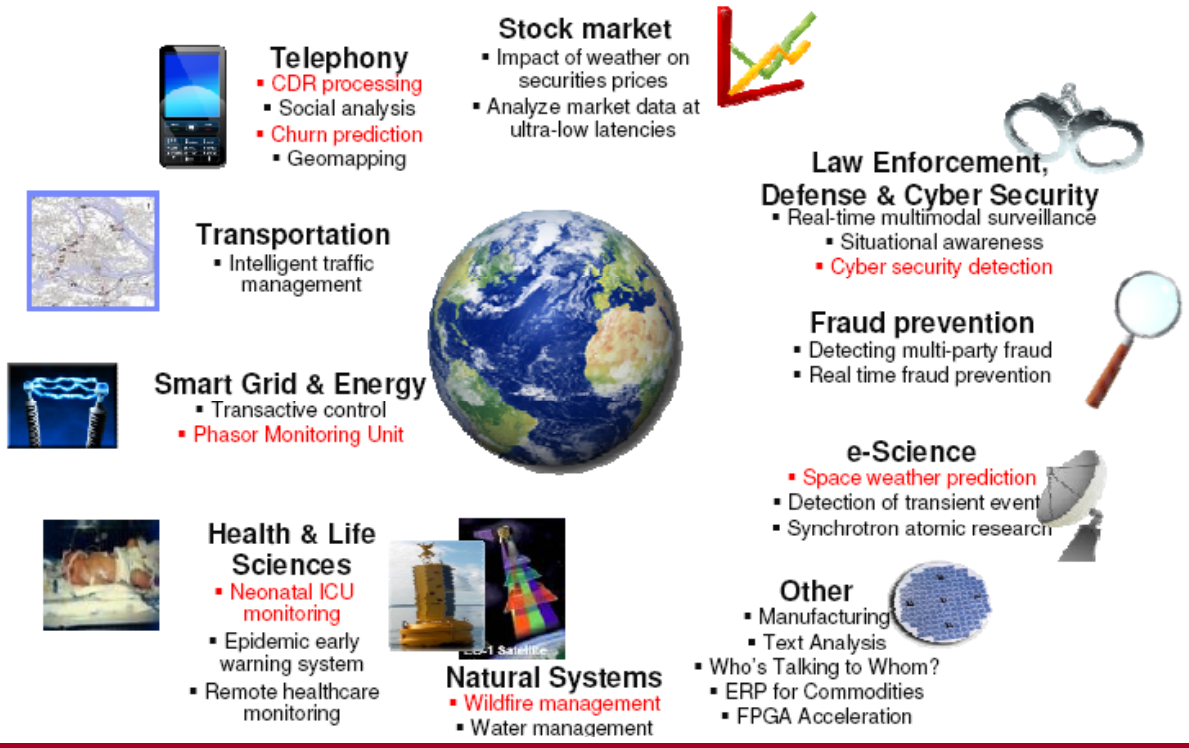
- Modelli, strutture e tipologie di dato
- Data Collection
- Data Processing
- Inferenza e attachment automatico di metadata
- Information Sharing
- Information Maintenance
- Information Archiving
- Information Protection

All-Data Service and Management

Piattaforme integrate e innovative per All-data service richiedono nuovi meccanismi, algoritmi, protocolli e strategie da investigare, anche in aree tecnologiche nuove

- ***Nuove metodologie, processi e strumenti***
- ***Causality/correlation sui dati***
- ***Stream processing***
- Visual Analytics
- ***Real-time Analytics***
- ***Sicurezza e data quality (QoS)***
- Mantenimento e archiviazione dei dati
- Nuove architetture hardware e di storage

Big Data: Aree Applicative



Sicurezza e Law Enforcement (1)

- Video sorveglianza, comunicazioni, recording di chiamate (sia voce che email), ..
- Milioni di messaggi al secondo con ***bassa densità di dati critici***
- Identificazione di ***pattern e relazioni*** fra sorgenti di informazioni dal volume imponente e molto numerose

Ad es. US Government e IBM per high-performance analytics ad ***alta scalabilità*** su ***multimedia stream "in motion" di tipo eterogeneo***

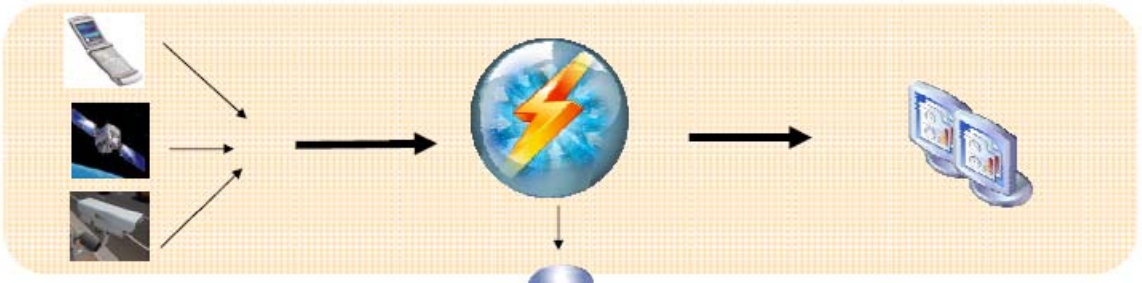
Sicurezza e Law Enforcement (2)

Government and Law Enforcement: e911 Support

Data from 911 calls, satellite feeds, imagery from city traffic cameras

Streams defines the geo spatial location of the call by running powerful analytics in real time using satellite communication link and draws in city camera feeds from around the area

Real time support for 911 dispatcher and field personnel

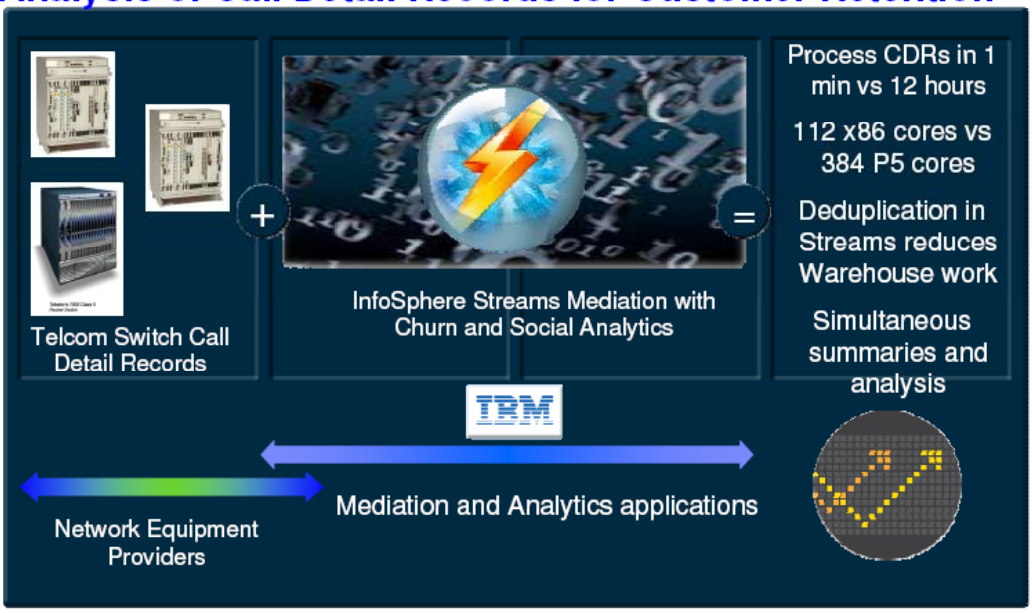


Data stored for future auditing and evidence requirements

Forme Avanzate di Customer Relationship Management (1)

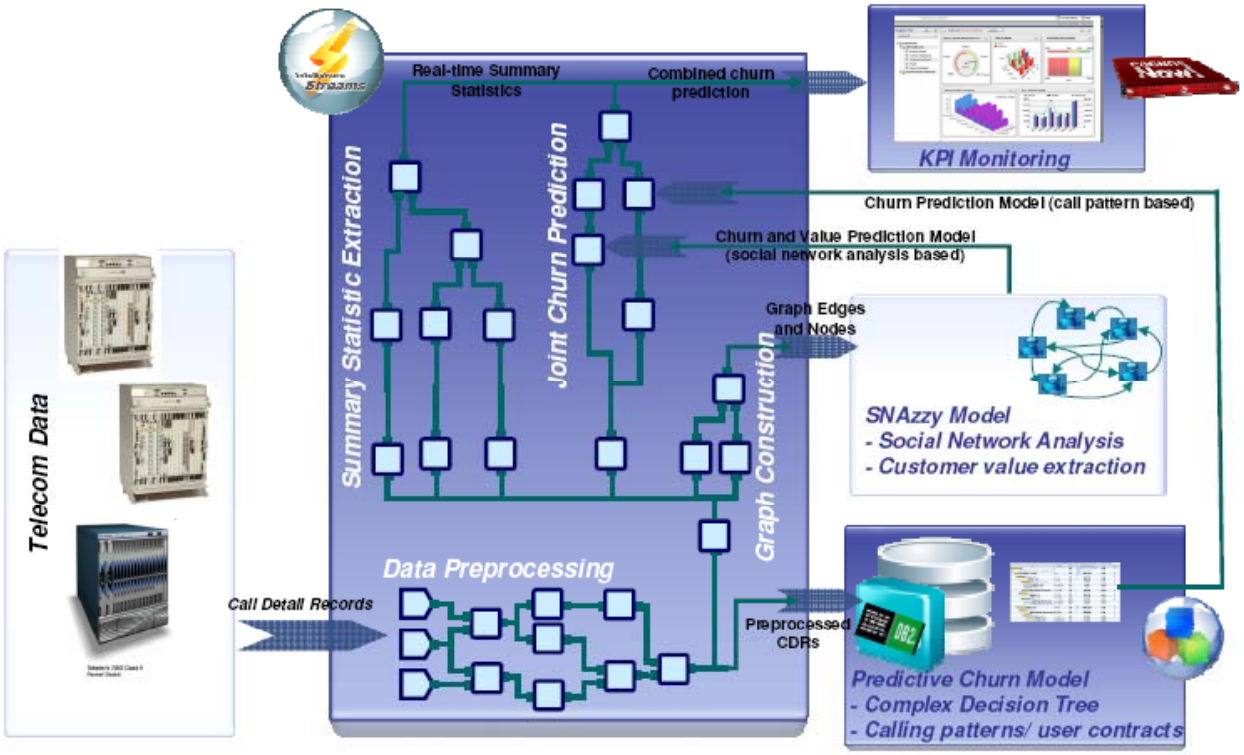
Non solo riconoscimento di ***problemi comuni*** e reperimento info per ***determinati utenti/classi di utenti***, ma anche ***trend emergenti e "sentiment analysis"***

Analysis of Call Detail Records for Customer Retention



Forme Avanzate di Customer Relationship Management (2)

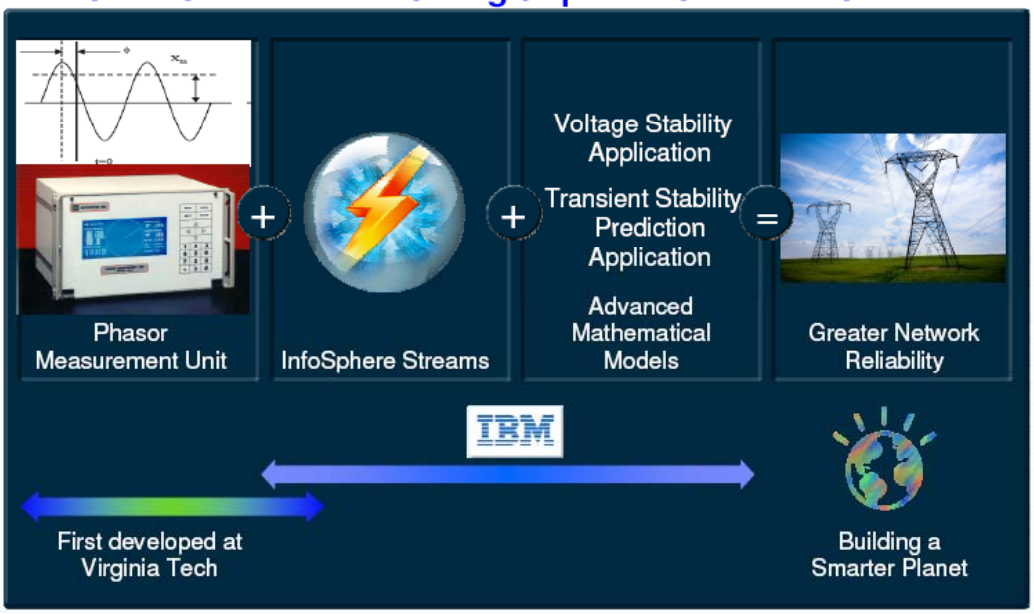
Telephony Architecture



Integrazione e Ottimizzazione Smart Grid (1)

- Concetto emergente di ***smart grid***
- Anche microgenerazione di energia e difficoltà di storage

Smart Electric Grid Monitoring to prevent black outs

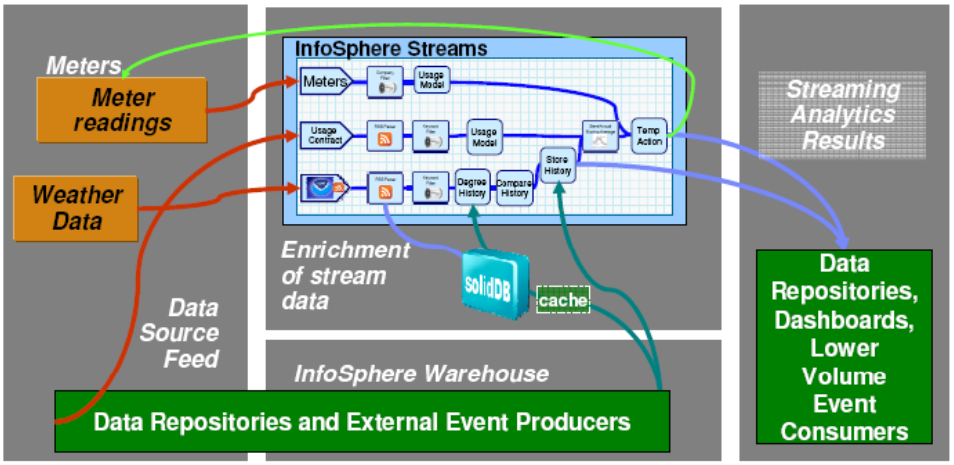


Integrazione e Ottimizzazione Smart Grid (2)

Importanza della possibilità di inserire ***metadata di correlazione*** sui flussi in elaborazione

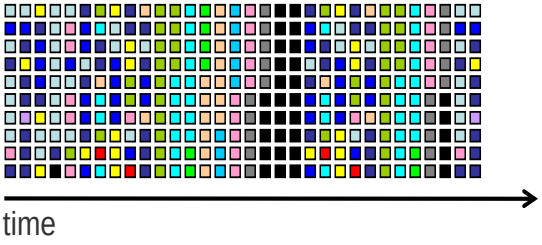
Smarter Utilities

- Analysis of data from multiple sources
- Real-time decisions – lower voltage on hot days

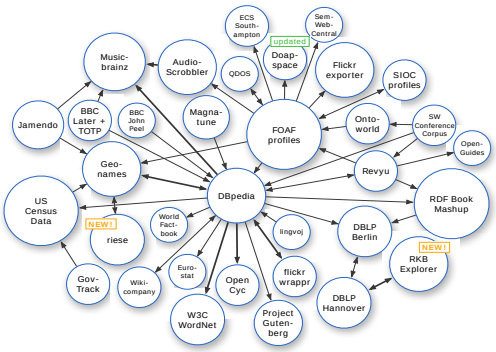


Molte Aree Tecnologiche Coinvolte

Enormi moli di dati, quasi sempre eterogenei, e spesso in ***real-time streaming*** che devono essere monitorati, con gestione spesso ***urgente di situazioni critiche o comunque di interesse***



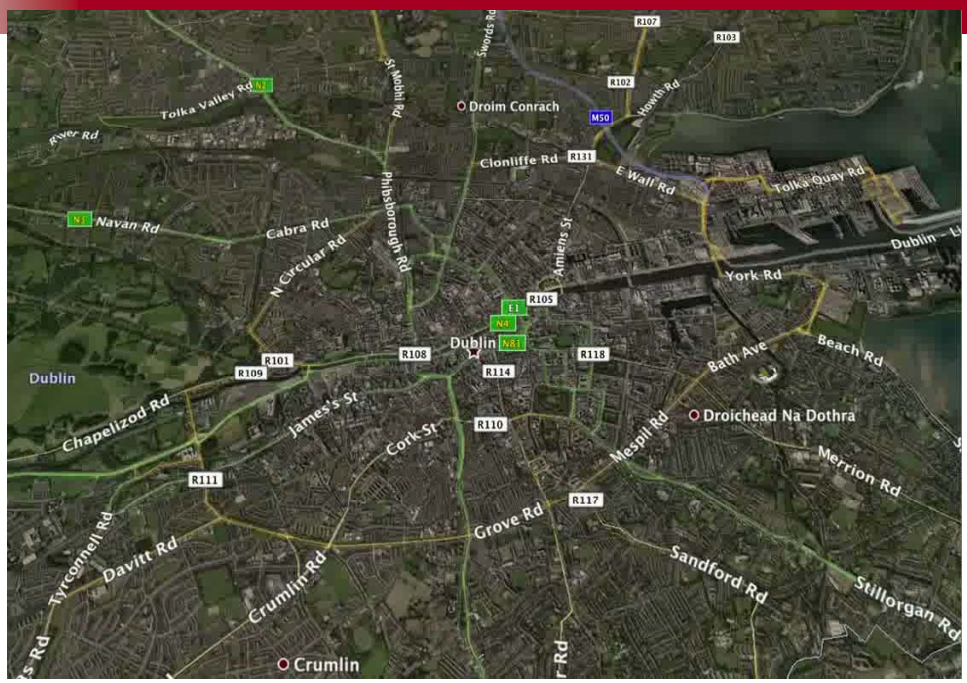
- Acquisizione dati da sensori
  - Eterogeneità
- Tecnologie data streaming
  - Dati statici/dinamici
  - Identificazione e gestione
- Semantic Web e tecnologie sem. lightweight
  - Linked and Open Data



Data Streaming from a Smart City

Sorgenti:

- Biciclette
- Noise control
- Pollution control
- Punti di interesse
- Bus
- Video-camere

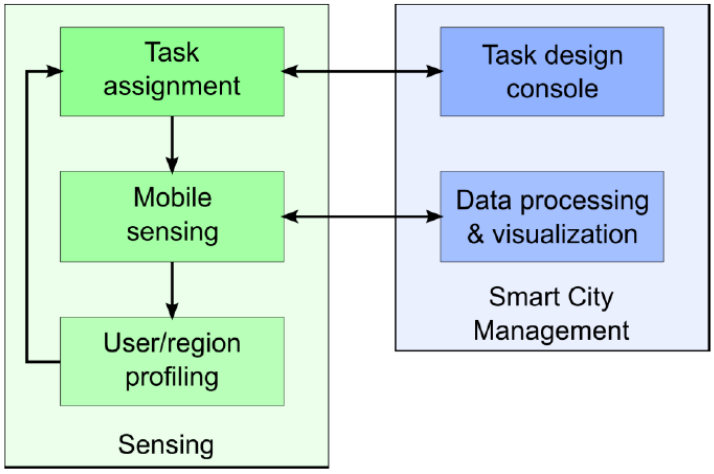


Analisi di alcune tipologie di sorgenti dati dalla città di ***Dublino***, per monitorare ***con focus variabile alcuni punti di interesse via "selected activated cameras"***

ParticipAction: Crowdsensing

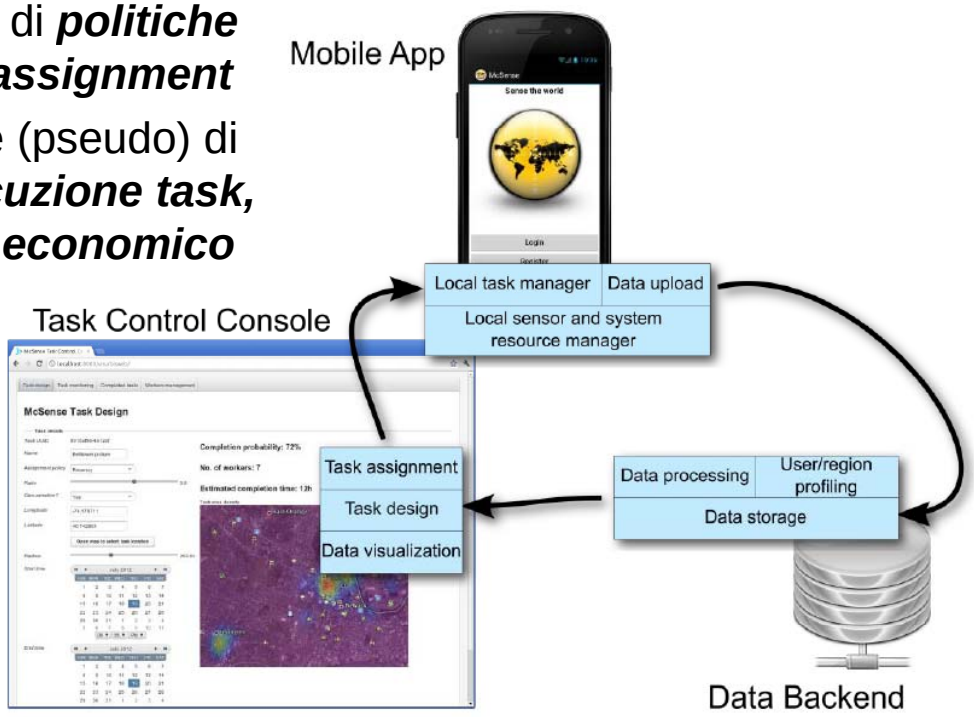
- Collaborazione con NJIT
- Disponibilità di buon gruppo di terminali Android e utenti per sperimentazione (300)

- Monitoring e crowdsensing per smart city
- ***Assegnamento "smart" di task partecipativi***, anche con micro-incentivi economici



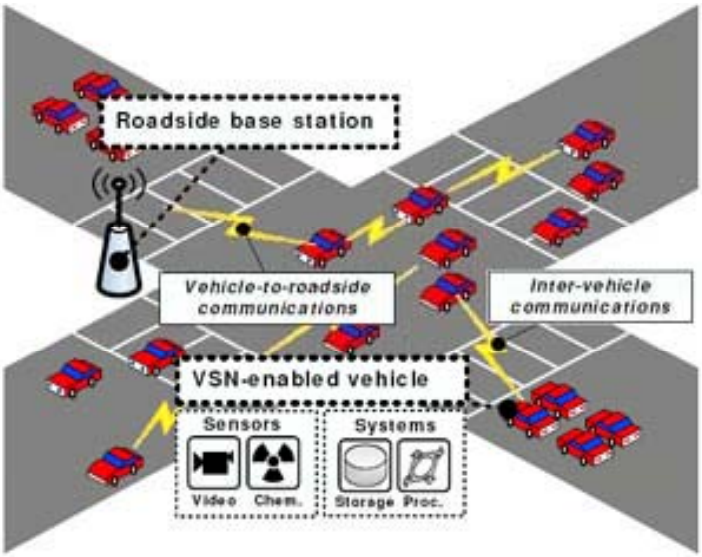
ParticipAction: Task Assignment

- Determinazione e sperimentazione di ***politiche smart per task assignment***
- Ottimizzazione (pseudo) di ***affidabilità esecuzione task, latenza e costo economico***



Vehicular Traffic Management

Automobili sono esempio potente di ***sensori mobili autonomi*** e che si possono ***coordinare*** in modo ***lazy*** sfruttando comunicazioni wireless



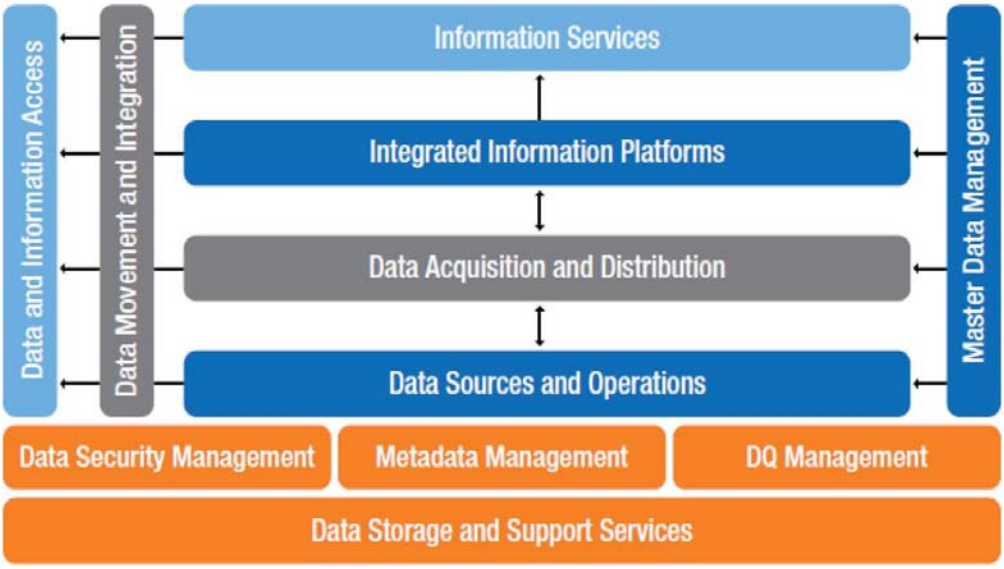
- Automobili operano ***sensing opportunistico*** nell'ambiente urbano e mantengono dati localmente
- ***Disseminazione collaborativa di metadati*** basata su decisioni locali
- Possibilità di ***comportamenti emergenti*** per soddisfare ***requisiti application-specific*** (ad es. completezza query, tempo risposta, overhead, …)

Vedi progetto EU FP7 COLOMBO

Sforzi di Standardizzazione Industriale

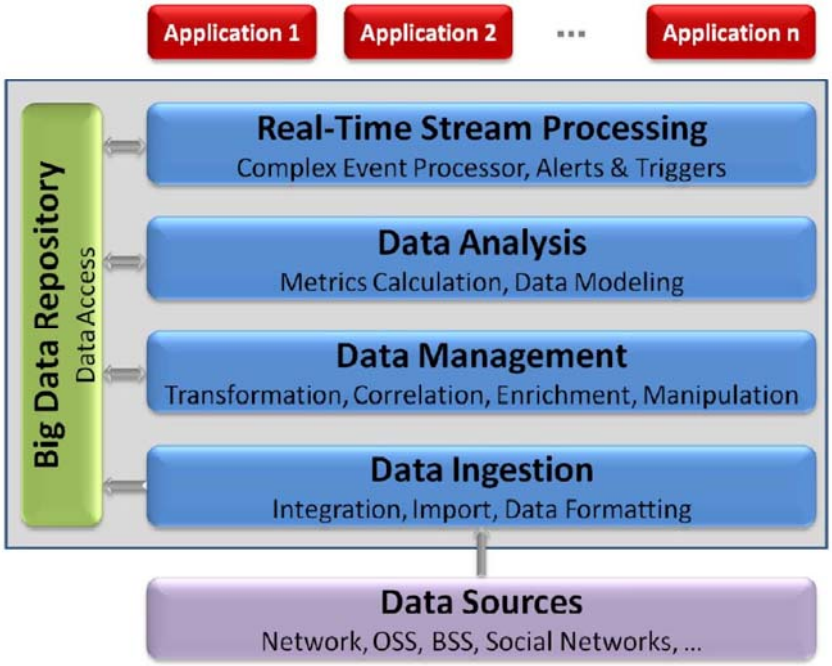
Standardizzazione industriale almeno delle ***architetture emergenti*** nelle piattaforme di supporto

Open Data Center Alliance (ODCA) for Information as a Service (InfoaaS)



Sforzi di Standardizzazione Industriale

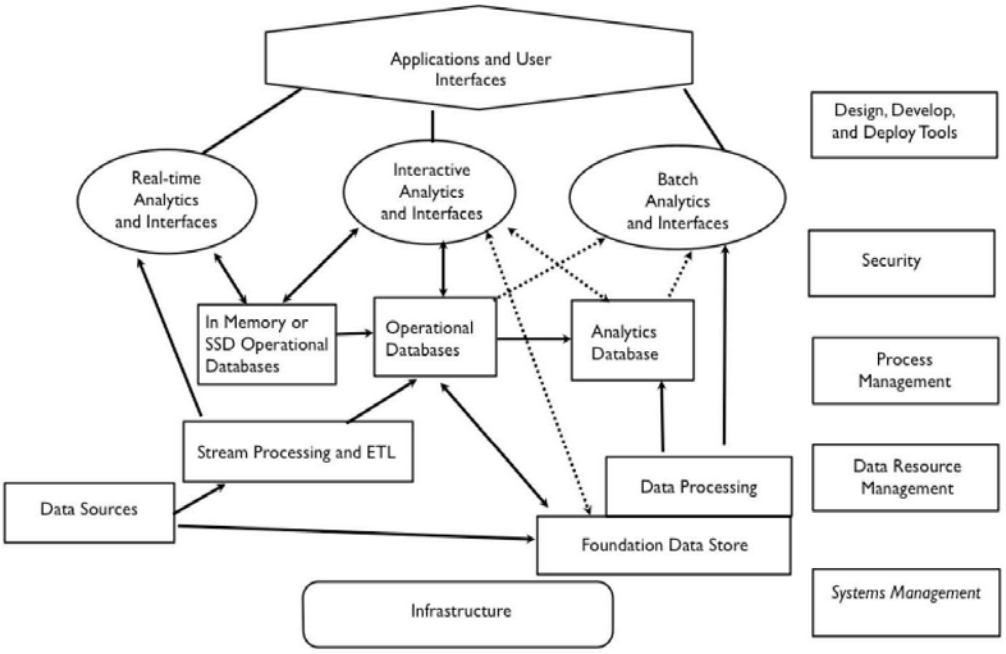
- Architettura TMF per Big Data Analytics
- Strumenti innovativi di analytics per accesso rapido ed efficiente



Sforzi di Standardizzazione Industriale

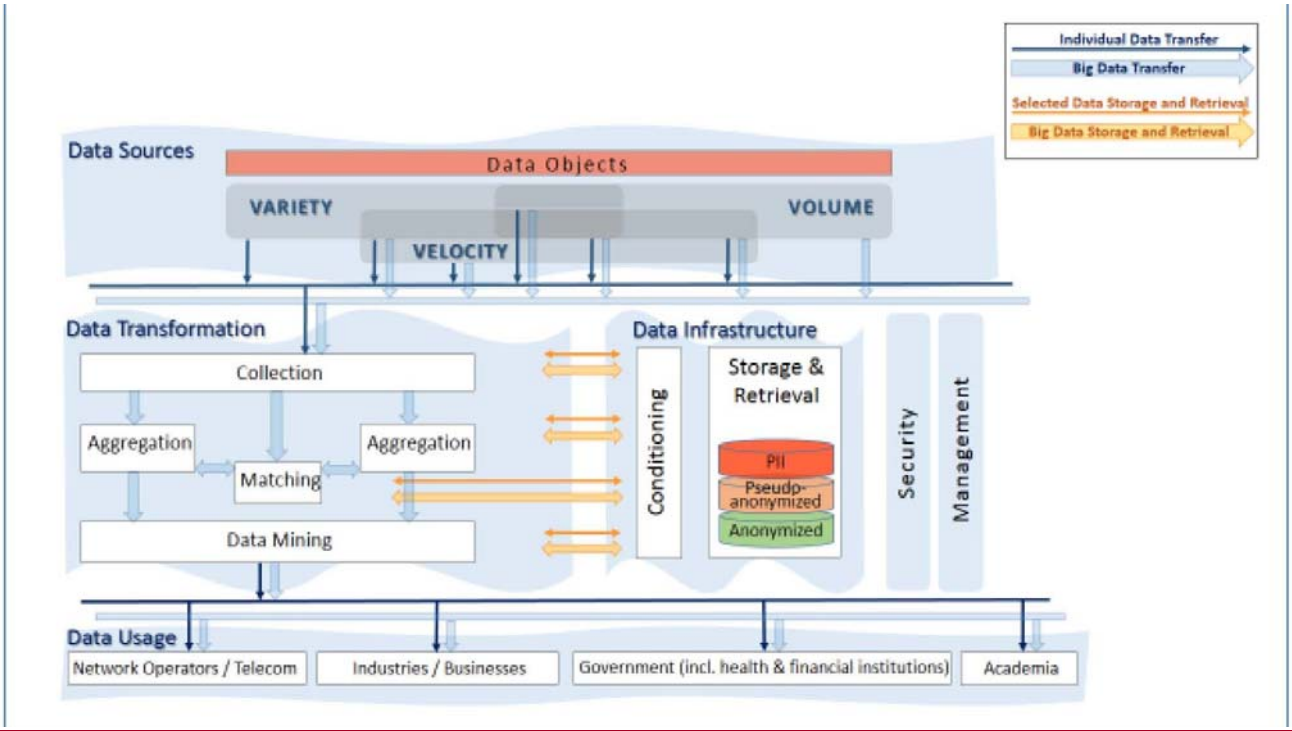
Architettura di riferimento ***NIST***

Distinzione fra real-time, interactive e batch



Sforzi di Standardizzazione Industriale

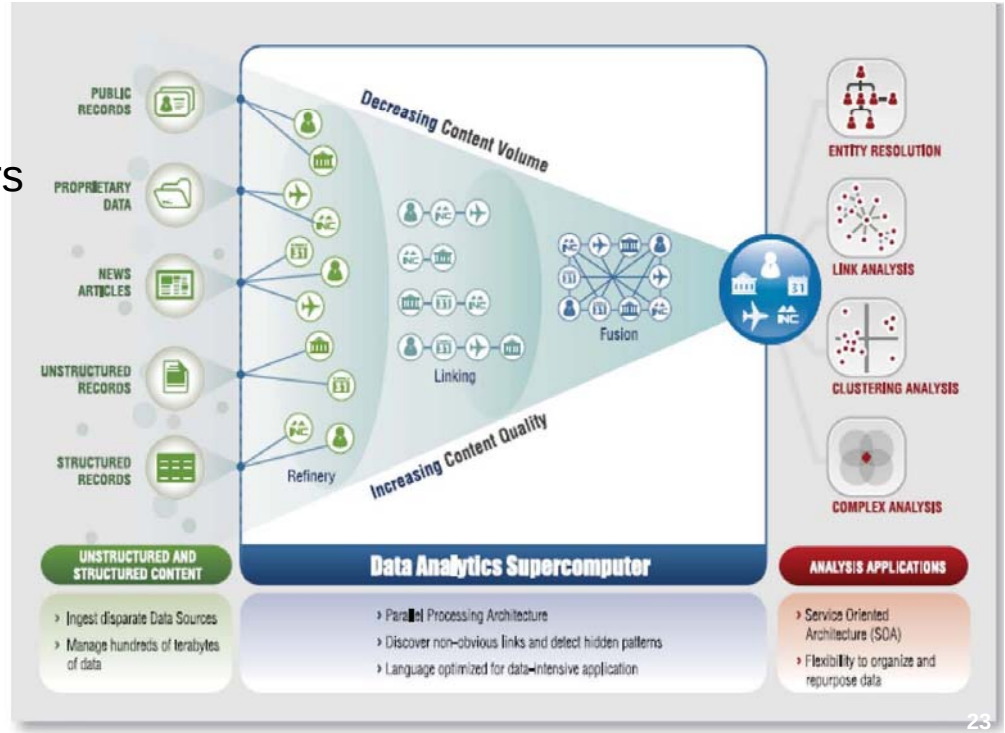
Ovviamente anche Microsoft…



Sforzi di Standardizzazione Industriale

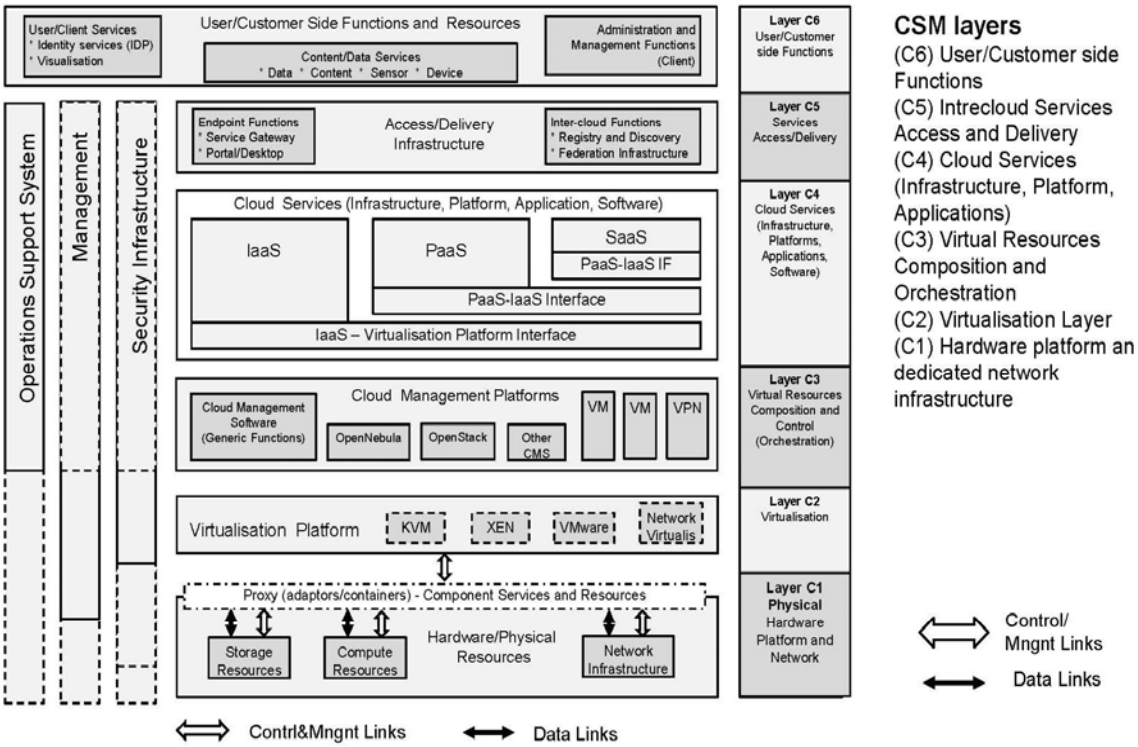
Architettura Lexis per Data Analytics Supercomputers

Idea di ***riduzione volumi e incremento qualità***



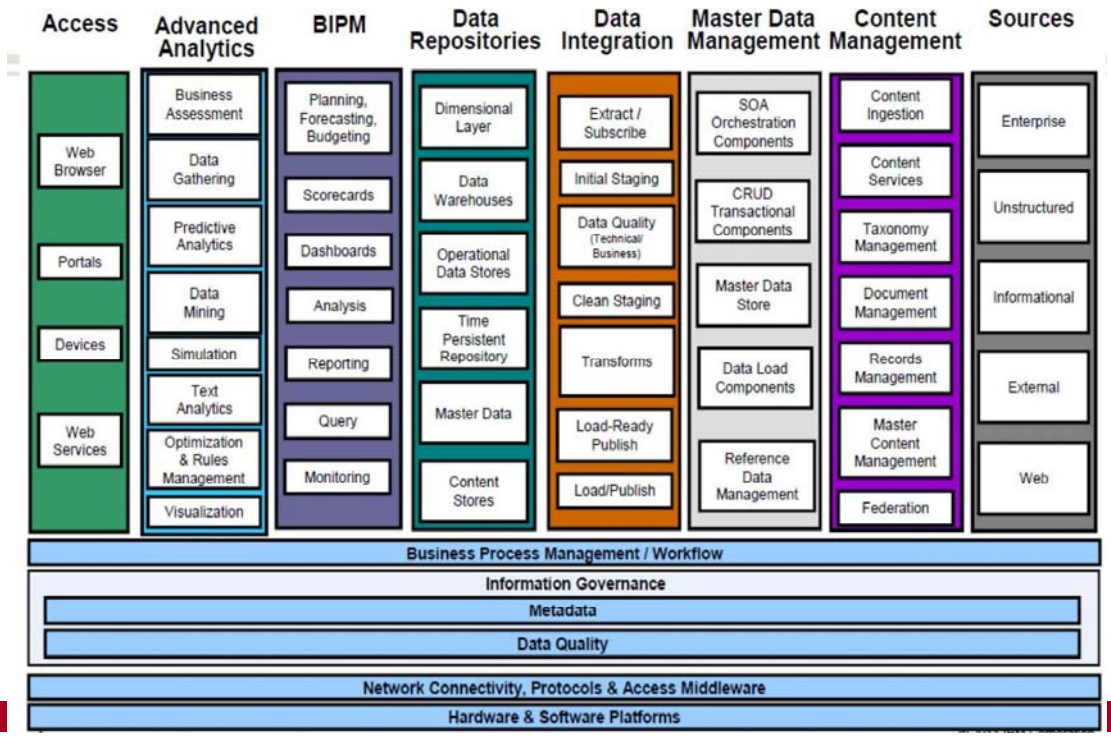
Sforzi di Standardizzazione Industriale

Ovviamente anche ***cloud integration***…



Sforzi di Standardizzazione Industriale

E ovviamente anche ***IBM***… - IBM Reference Architecture

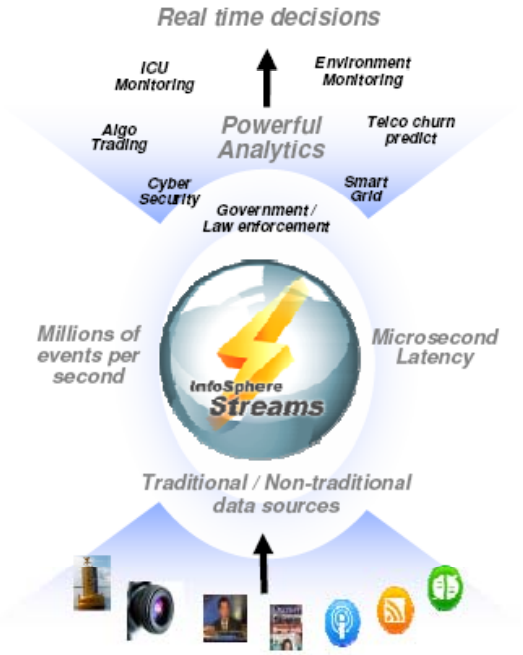


IBM InfoSphere Streams

IBM InfoSphere Streams v2.0

A platform for real-time analytics on BIG data

- **Volume**
  - Terabytes per second
  - Petabytes per day
- **Variety**
  - All kinds of data
  - All kinds of analytics
- **Velocity**
  - Insights in microseconds
- **Agility**
  - Dynamically responsive
  - Rapid application development



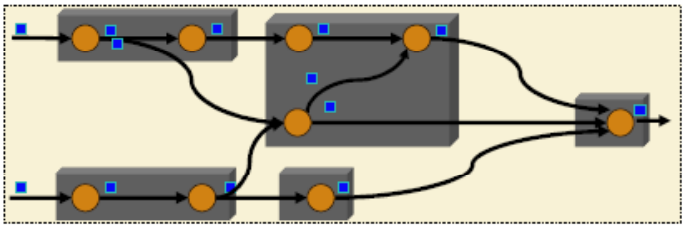
Perché InfoSphere Streams

Differenze rispetto a sistemi di data analytics più tradizionali (non big-data-oriented):

1. Orientato ad applicazioni che richiedono ***on-the-fly processing, filtering e analisi di flussi*** di dati
   - Sensori (ambientali, industriali, video sorveglianza, GPS, …)
   - Log file di server network/Web/application
   - Dati di transazioni ad alto rate: transazioni finanziarie, chiamate telefoniche, …
2. ***Criteri***
   - Messaggi processati in ***isolamento o in finestre limitate***
   - Sorgenti includono ***dati non tradizionali*** (spaziali, immagini, …)
   - Sorgenti ***eterogenee*** in termini di connettività, datarate, requisiti di processamento, … con problematiche di integrazione
   - Datarate e volumi richiedono risorse computazionali su ***nodi di processamento multipli,*** troppo grandi per ***approcci store-and-mine***
   - Tempi di risposta con ***latenze dell'ordine sub-ms***

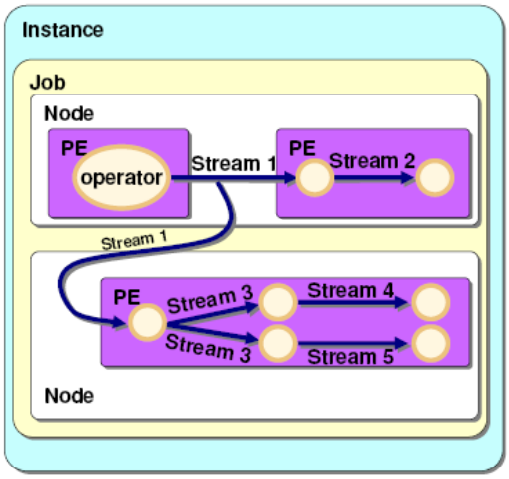
Scalable Stream Processing

- ***Modello di programmazione per definire grafi dataflow*** costituiti da datasource (input), operatori e sink (output)
- Controllo più o meno automatico per ***fusione di operatori in Processing Element*** (PE)
- ***Infrastruttura*** per supportare composizione e runtime di applicazioni stream processing ***scalabili*** a partire da questi componenti
- Deployment e supporto runtime su nodi x86/PowerPC, anche ad alte prestazioni (ad es. IBM Blade)



Applicazione Streams

Applicazione Streams come ***grafo diretto, possib. ciclico***, dioperatori connessi da streams



Istanza

- Istanziazione runtime di InfoSphere Streams che esegue su uno o più nodi
- Collezione di componenti e servizi

PE

- Unità di esecuzione fondamentale che è eseguita da una istanza
- Può incapsulare un singolo operatore o diversi operatori "fused"

Job

- Una applicazione Streams di cui è fatto il deployment su una istanza
- Consiste di uno o più PE

Streams: Development View

Operatore

Building block fondamentale per Streams Processing Language. Operatori processano dati chiamati Streams e possono generare nuovi stream

Stream

Una ***sequenza infinita di tuple strutturate***. Possono essere consumate da operatori o in modo singolo o attraverso la definizione di una finestra

Tupla

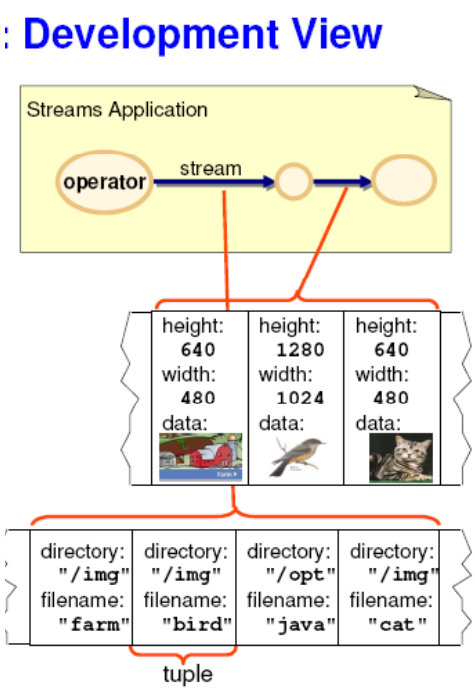
Lista strutturata di attributi e dei lori tipi

Stream type

Specifica del nome e del tipo per ogni attributo in una tupla

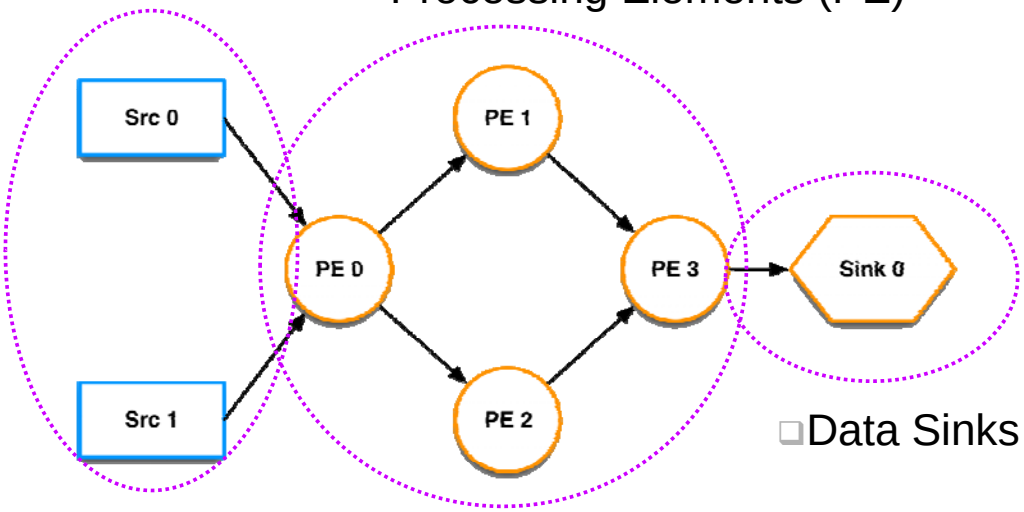
Finestra

- ***Gruppo finito e sequenziale*** di tuple in un flusso
- Basata su contatori, tempo, valore di attributi o punctuation mark



Streams Processing Model: Flow Graph

- Data Sources
- Operators / Processing Elements (PE)



- Data Sinks

Ambiente di Sviluppo e Runtime Integrato

IBM InfoSphere Streams

Agile Development Environment

- Eclipse IDE
- Streams Live Graph
- Streams Debugger

Distributed Runtime Environment

- Clustered runtime for near-limitless capacity
- RHEL v5.3 and above, CentOS v6.0 and above
- x86 & Power multicore hardware
- Ethernet & InfiniBand

Sophisticated Analytics with Toolkits & Adapters

- Toolkits
- Database
- Mining
- Financial
- Standard
- Internet
- Big Data (HDFS)
- Text
- User-defined
- Over 50 samples

Le Origini del Modello…

Da dove traggono origine ***modello e successo*** dell'approccio?

Probabilmente il "padre storico" di impatto industriale è il progetto ***Apache Hadoop***

- ***Framework open source*** di larga scala, con Yahoo! Come principale contributor
- Dedicato a finalità di ***processing scalabile, distribuito e data-intensive***
- Migliaia di nodi e PB di dati
- Supporta applicazioni sotto free license
- 3 sottoprogetti principali
  - Hadoop Common (package di facilities comuni)
  - ***Hadoop Distributed File System (HDFS)*** – high throughput
  - ***MapReduce*** – framework per processing distribuito di grandi insiemi di dati su cluster

MapReduce

MapReduce è ***modello di programmazione e framework software*** sviluppato originariamente da Google (paper 2004)

Obiettivo: semplificare il processamento di enormi moli di dati ***in parallelo su cluster di grandi dimensioni di commodity hw***, in modo affidabile e fault-tolerant

Processamento deve avvenire su:

- ***Dati NON STRUTTURATI*** (filesystem)
- Dati strutturati (db)

Hadoop Distributed File System (HDFS)

Prende ispirazione da ***Google file system***

- ***Scalabile, distribuito, portabile, scritto in Java per framework Hadoop***
- HDFS può essere parte di cluster Hadoop o file system distrbuito stand-alone general-purpose
- HDFS cluster è costituito da:
  - NameNode che gestisce i ***metadata*** del file system
  - DataNode che memorizzano i veri dati
- Memorizza file di grandi dimensioni in ***blocchi distribuiti*** sul cluster
- Affidabilità e fault-tolerance tramite ***replicazione su nodi multipli***
- Progettato specificamente per deployment su hw low-cost

Hadoop può lavorare su qualsiasi file system distribuito ma sfrutta ***conoscenza di località per ottimizzazione***, quindi HDFS particolarmente adatto

Hadoop Cluster

Un tipico cluster Hadoop integra ***funzionalità MapReduce e HDFS***

- ***Architettura master/slave***
- Master contiene
  - Job tracker (MapReduce –responsabile scheduling dei job task, monitoraggio slave, ri-esecuzione job con fallimenti)
  - Task tracker (MapReduce)
  - NameNode (HDFS)
  - DataNode (HDFS)
- Nodi slave includono
  - Nodo Task tracker (MapReduce – esegue i task sotto coordinamento del master)
  - DataNode (HDFS)

MapReduce (1)

Codice usualmente scritto in Java, altri linguaggi supportati tramite Hadoop Streaming API

Due passi fondamentali:

Map step

- Nodo master riceve input del problema e lo divide in ***sotto-problemi più piccoli***, distribuiti verso i nodi worker. ***Quando e come è possibile?***
- Nodi worker possono farlo a loro volta ***(struttura gerarchica ad albero multi-livello)***
- Un worker risolve problema "piccolo" e riporta sotto-risultato al master

Reduce Step

- Nodo master ***raccoglie risposte ai sottoproblemi*** e li combina in modo predefinito per ottenere la risposta complessiva

MapReduce (2)

Funzionalità core e come è gestito il flusso dati fra gli step di Map e Reduce:

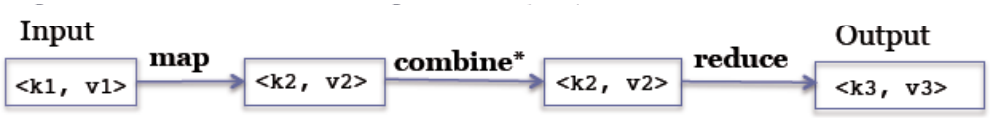
- ***Input reader*** – divide input in chunk di ***misura appropriata***, che vengono assegnati a una funzione Map
- ***Funzione Map*** – mappa file data ***verso coppie <key, value>*** più piccole e di utilizzo intermedio
- ***Funzione Partition*** – trova il ***reducer appropriato*** data la chiave key
- ***Funzione Compare*** – input per Reduce è preso dall'output intermedio di Map e riordinato in accordo a Compare
- ***Funzione Reduce*** – prende valori intermedi e processa soluzione parziale restituita al framework
- ***Output writer*** – scrive risultati su file di output

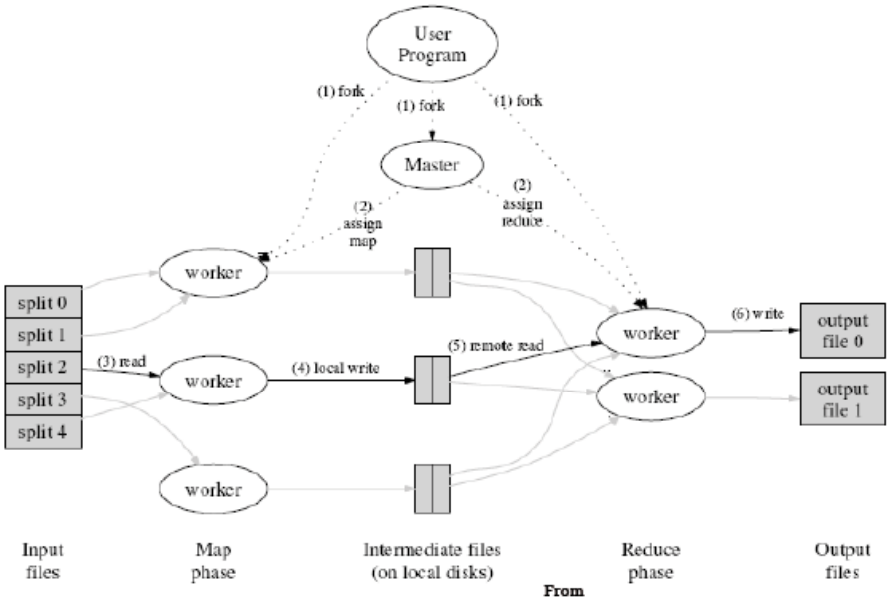
MapReduce (3)

Funzionalità core e come è gestito il flusso dati fra gli step di Map e Reduce (continua):

- ***Un Job MapReduce controlla l'esecuzione***
  - Divide dataset di input in ***chunk indipendenti***
  - Chunk indipendenti sono processati da task Map ***in parallelo***
- Il framework riordina gli output dei Map
- Un task MapReduce raccoglie l'output e svolge il ruolo di reduce & combine
- Sia input che output del job sono memorizzati nel file system integrato in Hadoop
- ***Il framework gestisce tutte le problematiche di scheduling***
  - Monitoraggio e ri-esecuzione di task con fallimenti/guasti

MapReduce (4)





From http://code.google.com/edu/parallel/mapreduce-tutorial.html

MapReduce (5)

Esempio semplicissimo: WordCount per contare occorrenze di OGNI PAROLA su un set di file di ingresso

2 file di ingresso:
file1 = “hello world hello moon”
file2 = “goodbye world goodnight moon”

3 operazioni:
Map
Combine
Reduce

MAP

First map:
```
< hello, 1 >
< world, 1 >
< hello, 1 >
< moon, 1 >
```

Second map:
```
< goodbye, 1 >
< world, 1 >
< goodnight, 1 >
< moon, 1 >
```

COMBINE

First map:
```
< moon, 1 >
< world, 1 >
< hello, 2 >
```

Second map:
```
< goodbye, 1 >
< world, 1 >
< goodnight, 1 >
< moon, 1 >
```

REDUCE
```
< goodbye, 1 >
< goodnight, 1 >
< moon, 2 >
< world, 2 >
< hello, 2 >
```

MapReduce Scheduling (1)

Hadoop scheduling di default per i job è ***FIFO***

Alternative = ***capacity e fair***

Capacity scheduler

- Sviluppato da Yahoo!
- Job ***sottomessi a code e prioritizzati (priorità statica o dinamica?)***
- Code con ***allocazione di una frazione*** della capacità totale delle risorse disponibili
- ***Risorse libere*** sono allocate alle code in aggiunta alla loro capacity totale nominale
- ***Nessuna preemption***

MapReduce Scheduling (2)

Fair scheduler

- Sviluppato da Facebook
- Migliora tempi di risposta per ***job "piccoli"***
- Job sono raggruppati in ***insiemi detti Pool***
- Ad ogni Pool è assegnata una ***quota minima garantita***
- La capacità in eccesso è divisa fra i job
- A default i job non categorizzati (uncategorized) vanno nel Pool di default.

I Pool devono specificare il loro ***numero minimo di risorse desiderate di tipo map e resource***, e un limite sul numero dei loro job in esecuzione

Limiti di Hadoop e altri Approcci

- Come effettuare la determinazione di ***chunk indipendenti e sotto-problemi indipendenti***?
- Anche MapReduce ***più adatto a batch processing***, meno a stream processing
- Approccio non "dramatically novel" – vedi tradizione del calcolo distribuito e parallelo

Anche di conseguenza, estremo fermento nell'ambito piattaforme di stream processing. Non solo IBM InfoSphere Streams:

- Storm - http://www.slideshare.net/nathanmarz/storm-11164672
- Apache S4 - http://www.slideshare.net/alekbr/s4-stream-computing-platform
- YARN (Yet Another Resource Negotiator) per maggiore scalabilità su MapReduce - http://www.slideshare.net/AdamKawa/apache-hadoop-yarn-simply-explained
- Apache Samza (LinkedIn) - http://www.slideshare.net/blueboxtraveler/apache-samza

Appendice

Alcune attività di ricerca svolte nel settore Big Data e Distributed Stream Processing da UNIBO:

- ***Quasit*** (streaming graph con indicazioni/requisiti di qualità di servizio e integrazione con OMG DDS)
- Collaborazione con ***IBM Dublin Smarter Cities Lab***
- Collaborazione con ***UTA e Numerex***
- Bilanciamento e tradeoff dinamico di ***qualità di servizio e garanzie di fault tolerance tramite replicazione adattiva*** (vedi qualche dettaglio nel seguito…)

Fault-Tolerance in Distributed Stream Processing Systems (DSPS)

- Numero crescente di applicazioni hanno la necessità di gestire, trasformare e analizzare Big Data Stream in modo scalabile ed efficace:
  - Monitoring
  - Crowdsensing
  - Healthcare
  - Web 2.0
  - Transportation
  - Finance
- ***Applicazioni DSPS devono eseguire per sempre e senza interruzione***
  - ***Guasti sono sicuri*** (hardware o software)
  - Su larga scala, affidabilità peggiora (maggior numero di possibili punti di failure)
- ***Nostro claim: DSPS hanno bisogno di fronteggiare guasti efficientemente in dipendenza da requisiti application-specific***
  - Ad es. una soluzione di social messaging ha diversi requisiti di qualità di una applicazione per personal healthcare

Fault-Tolerance in DSPS

Consistency

- Mascherare gli effetti dei guasti sull'output applicativo
- Semantica di processamento tuple
- Gestione di componenti stateful
- Idealmente processamento delle tuple exactly-once

Availability

- Failure detection
- Replacement dei componenti con fallimenti, ad es. PE
- Idealmente fail-over istantaneo per evitare crescite di latenza

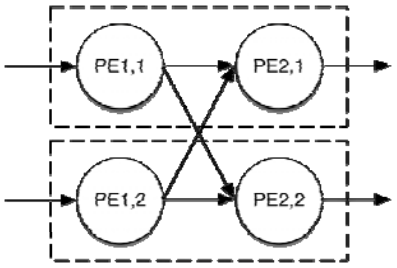
Cost

- Runtime cost: costo di mantenimento fault-tolerance in assenza di guasti
- Fail-over cost: costo per gestione fallimenti
- Ovviamente obiettivi di minimizzazione costi

Review di Tecniche di Fault-Tolerance

Active Replication

- ***Due repliche fisiche*** in esecuzione ***per ogni PE*** nel grafo di flusso
- Mantengono il loro ***stato consistente*** tramite processamento dello stesso input
- Solo la copia primaria emette le tuple di output verso le repliche downstream
- In caso di guasto del primario, il secondario può immediatamente fare take over (***latenza minima***)
- ***100% runtime overhead***
- ***Consistenza forte***
- Esempi: **Borealis**[1] o **Flux**[2]

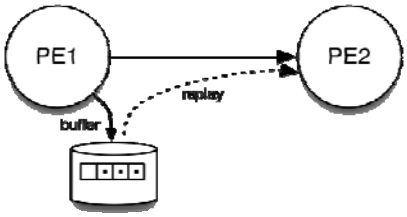


[1] M. Balazinska, et al.: Fault Tolerance in the Borealis Distributed Stream Processing System. ACM Trans. Database Syst. 33 (1), 2008

[2] M.A. Shah, et al.: Highly available, fault-tolerant, parallel dataflows. Proc. of the ACM SIGMOD Conference, 2004

Upstream Backup

- Tuple di output sono ***memorizzate "somewhere upstream"*** per ogni PE (ad es. dai predecessori o sorgenti)
- In caso di guasto, si avvia un PE con fresh state
- Tutte le ***tuple bufferizzate sono replayed***
- ***Latenza molto alta***: tuple devono essere replayed
- ***Basso runtime overhead*** (basso costo storage)
- ***Consistenza forte***
- Esempi: Storm[3] o Hwang et al.[4]

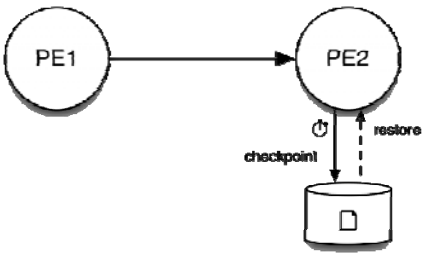


[3] Storm project Web Site. Available at: http://storm-project.net/, last visited 25 Sep. 2013

[4] J.-H. Hwang, et al.: High Availability Algorithms for Distributed Stream Processing. In Proc. of 21° ICDE Conf., Tokyo, Japan, 2005

Checkpointing

- ***Stato dei componenti è periodicamente checkpointed*** su storage permanente
- In caso di guasto, un PE è riavviato recuperando lo stato checkpointed
- ***Bassa latenza*** perché lo stato checkpointed è usualmente molto più piccolo dello storico tuple
- ***Basso runtime overhead***
- ***Possibile perdita di info*** in caso di guasti
- Esempi: Streams[5] o Apache S4[6]



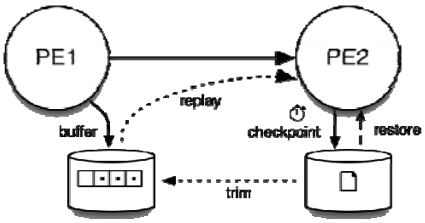
[5] G. Jacques-Silva, et al.: Language level checkpointing support for stream processing applications. In Proc. Of DSN'09 Conf., Lisbon, 2009

[6] L. Neumeyer, et al: S4: Distributed Stream Computing Platform. In Proc. Of. ICDMW'10 Conf., Sydney, Australia, 2010

Review di Tecniche di Fault-Tolerance

Checkpointing + Upstream Backup

- ***Stato di componenti è periodicamente checkpointed*** su storage permanente
- ***Tuple "non afferenti" ai checkpoint sono backed up*** su nodi upstream
- ***Latenza maggiore*** rispetto a checkpointing ma minore rispetto a upstream backup puro
- ***Basso runtime overhead***
- ***Consistenza forte***
- Esempi: Apache Samza[7] o Google MillWheel[8]



[7] C. Riccomini: Samza: Real-time Stream Processing at Linkedin. International Software Dev. Conf. Qcon, San Francisco,. 2013
[8] T. Akidau, et al.: MillWheel: Fault-Tolerant Stream Processing at Internet Scale. Proc. of the VLDB Conf., Trento, 2013

Review di Tecniche di Fault-Tolerance

Trade-off fra

- ***Recovery time (latenza)***
  quanto tempo richiesto per una applicazione per tornare attiva e di nuovo funzionante?
- ***Garanzie di consistenza***
  quale effetto dei guasti sullo stato dei componenti e sull'output dell'applicazione?
- ***Costo Runtime***
  quale runtime overhead per la soluzione di fault-tolerance?

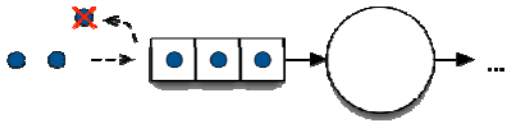
Technique	Latency	Consistency	Cost
Active Replication	Very Low	Strong	High
Upstream Backup	Very High	Strong	Low
Checkpointing	Low	Weak	Low
Checkpointing + Upstream Backup	High	Strong	Low

Gestione Cost-effective di Variazioni di Carico

- Molti flussi di dati real-world sono caratterizzati da ***variabilità della dinamica di input***, ad es. modifiche improvvise di datarate
  - Ad es. picchi di traffico in real-time monitoring di reti veicolari

- ***Carico si modifica fortemente in modo dinamico*** su sistemi di stream processing
  - Buffering di tuple alle porte input dei PE
  - Perdita di dati

- Altro ***trade-off*** fra ***latenza*** (buffering) e ***consistenza*** (tuple dropping)



Gestione di Variazioni di Carico

- **Over-provisioning**
  Allocare staticamente risorse sufficienti per gestire picchi di carico
- **Back-pressure**
  Rallentare componenti di processing (possibilmente fino alla sorgente)
- **Load-shedding**
  Invece di fare dropping casuale di tuple, cercare di scartare le tuple a ***minore importanza***
- **Dynamic PE Relocation**
  Muovere i componenti di processing da nodi overloaded a ***nodi sotto-utilizzati***

	Latenza	Perdita Dati	Costo
Over-provisioning	Bassa	Nessuna	Alto
Back-pressure	Alta	Nessuna*	Basso
Semantic Load Shedding	Bassa	Sì	Basso
Dynamic PE relocation	Dipende	Nessuna	Dipende

Load-Adaptive Active Replication (LAAR)

Punto aperto a livello di ricerca

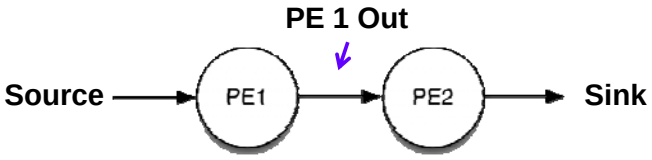
è possibile riutilizzare temporaneam le risorse normalm dedicate a fault-tolerance per la gestione efficace di picchi di traffico, mantenendo comunque un livello di servizio garantito?

- ***LAAR*** è una variante dinamica di tecniche di ***Active Replication***:
  Le repliche PE sono attivate/disattivate dinamicamente in dipendenza
  - Dal carico di sistema corrente
  - Dai requisiti di garanzie di consistenza
- Requisiti dello scenario target
  - ***Bassa latenza***, anche durante picchi di traffico
  - ***Trade-off dinamicamente modificabile*** fra garanzie di consistenza e costo runtime
  - Poter sostenere ***perdita di dati limitata e predicibile*** solo nel caso di guasti che occorrano durante picchi di traffico

LAAR in uno Scenario Iper-semplificato

Streaming processing graph:

- Pipeline di due PE con single-input single-output



- Descrizione sintetica della applicazione di streaming:

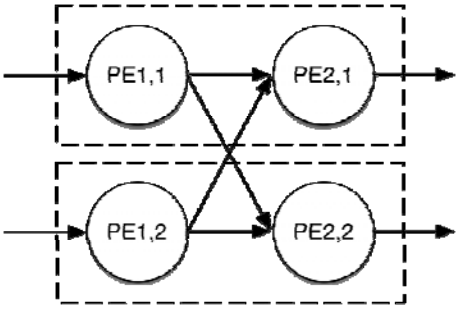
	PE 1	PE2
Selectivity	1	1
CPU per tuple	0.1 s/t	0.1 s/t

	Data Rate
Source	4 t/s
PE 1 Out	4 t/s

LAAR in uno Scenario Iper-semplificato

Deployment su due nodi con ***active replication***

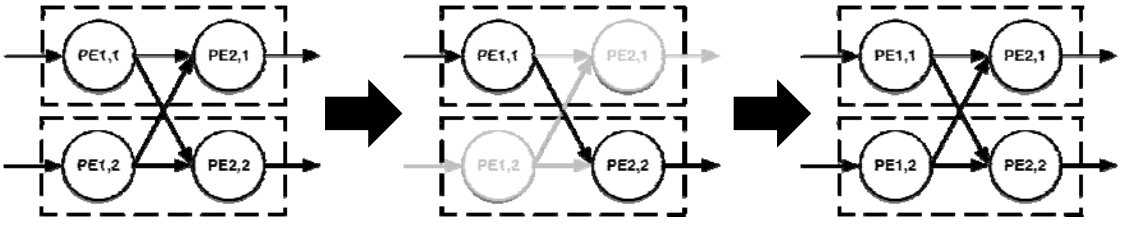
- Due copie di ogni PE su host differenti
- Ovviamente protegge da guasti singoli software o hardware

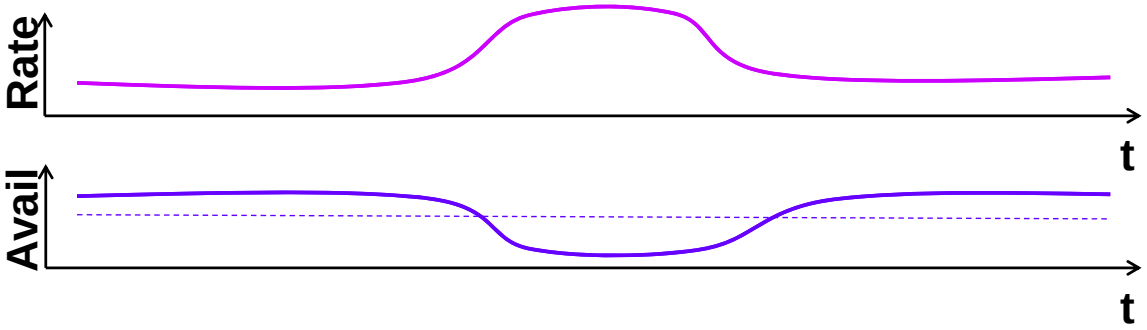


	PE 1	PE2
Selectivity	1	1
CPU per tuple	0.1 s/t	0.1 s/t

	Rate	Peak
Source	4 t/s	8 t/s
PE 1 Out	4 t/s	8 t/s

Disattivazione Dinamica delle Repliche in LAAR





LAAR è integrato in IBM InfoSphere Streams