Hálózatba kapcsolt erőforrás platformok és alkalmazásaik

Simon Csaba
TMIT
2018

Motivation

- Process lots of data
  - Google processed about 24 petabytes of data per day in 2009.
- **A single machine** cannot serve all the data
  - You need a distributed system to store and process **in parallel**
- Parallel programming?
  - **Threading** is hard!
  - How do you facilitate communication between nodes?
  - How do you **scale to more machines**?
  - How do you handle machine failures?

BIGDATA?

What is BigData?

"Big Data" exceeds the capacity of traditional analytics and information management paradigms across what is known as the 4 V's: Volume, Variety, Velocity, and Veracity

Veracity	Velocity	Variety	Volume
Uncertainty of Data	*Analysis of Streaming Data*	*Different Forms of Data*	*Scale of Data*
With exponential increases of data from unfiltered and constantly flowing data sources, data quality often suffers and new methods must find ways to "sift" through junk to find meaning	The speed at which data is generated and used. New data is being created every second and in some cases it may need to be analyzed just as quickly	Represents the diversity of the data. Data sets will vary by type (e.g. social networking, media, text) and they will vary how well they are structured	Reflects the size of a data set. New information is generated daily and in some cases hourly, creating data sets that are measured in terabytes and petabytes

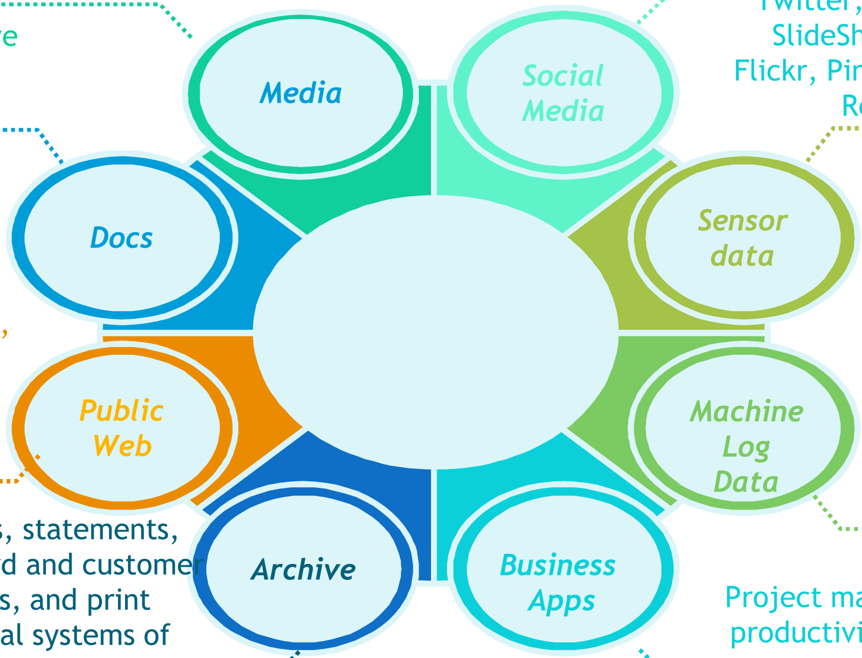
The Promise of Big Data

Even more important than its definition is what Big Data promises to achieve: *intelligence in the moment.*

	Traditional Techniques & Issues	***Big Data Differentiators***
Vera city	Å**Does not account for biases, noise and abnormality in data**	ÅData is stored, and **mined meaningful to the problem being analyzed** Å**Keeps data clean and processes to keep 'dirty data' from accumulating in your systems**
Velo city	Å**No real time analysis**	**In real-time:** Å**Dynamically analyze data** ÅConsistently integrate new information ÅAuto deletes unwanted to ensure **optimal storage**
Var iety	Å**Compatibility issues** ÅAdvanced analytics **struggle with non-numerical data**	ÅFrameworks **accommodate varying data types and data models** Å**Insightful analysis with very few parameters**
Vol.	ÅAnalysis is limited to **small data sets** ÅAnalyzing large data sets = **High Costs & High Memory**	ÅScalable for **huge amounts of multi-sourced data** ÅFacilitation of **massively parallel processing** Å**Low-cost data storage**

Types of Big Data

Variety is the most unique aspect of Big Data. New technologies and new types of data have driven much of the evolution around Big Data.



Images, videos, audio, Flash, live streams, podcasts, etc.

Twitter, Linkedin, Facebook, Tumblr, Blog, SlideShare, YouTube, Google+, Instagram, Flickr, Pinterest, Vimeo, WordPress, IM, RSS, Review, Chatter, Jive, Yammer, etc.

XLS, PDF, CSV, email, Word, PPT, HTML, HTML 5, plain text, XML, JSON, etc.

Medical devices, smart electric meters, car sensors, road cameras, satellites, traffic recording devices, processors found within vehicles, video games, cable boxes, assembly lines, office building, ….

Government, weather, competitive, traffic, regulatory, compliance, health care services, economic, census, public finance, stock, OSINT, the World Bank, SEC/Edgar, Wikipedia, IMDb, etc.

Event logs, server data, application logs, business process logs, audit logs, call detail records (CDRs), mobile location, mobile app usage, clickstream...

Archives of scanned documents, statements, insurance forms, medical record and customer correspondence, paper archives, and print stream files that contain original systems of record between organizations and their customers

Project management, marketing automation, productivity, CRM, ERP content management system, HR, storage, talent management, procurement, expense management Google Docs, intranets, portals, etc.

"Single sources of data are no longer sufficient to cope with the increasingly complicated problems in many policy arenas." [1]

Big data "is not notable because of its size, but because of its relationality to other data. Due to efforts to mine and aggregate data, Big Data is fundamentally networked."[2]

(1) M. Milakovich, "Anticipatory Government: Integrating big data for Smaller Government", in Oxford Internet Institute "Internet, Politics, Policy 2012" Conference, Oxford, 2012
(2) D. Boyd and K. Crawford, "Six Provocations for big data," in A Decade in Internet Time: Symposium on the Dynamics of the Internet and Society, 2011

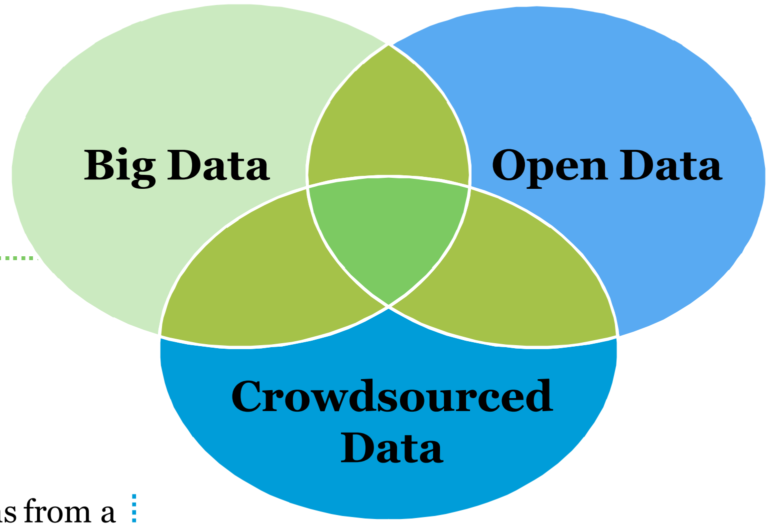
Why is BigData valuable?

We have identified 5 key areas where Big Data is uniquely valuable:

Accessibility to Data	Enhanced visibility of relevant information and better transparency to massive amounts of data. Improved reporting to stakeholders.
Decision Making	Next generation analytics can enable automated decision making (inventory management, financial risk assessment, sensor data management, machinery tuning).
Marketing Trends	Segmentation of population to customize offerings and marketing campaigns (consumer goods, retail, social, clinical data, etc).
Performance Improvement	Exploration for, and discovery of, new needs, can drive organizations to fine tune for optimal performance and efficiency (employee data).
New Business Models/Services	Discovery of trends will lead organizations to form new business models to adapt by creating new service offerings for their customers. Intermediary companies with big data expertise will provide analytics to 3rd parties.

Not to be confused with…

Structured, semi-structured or unstructured information distinguished by one or more of the four "V"s: Veracity, Velocity, Variety, Volume.



Public, freely available data

Data collected through contributions from a large number of individuals.

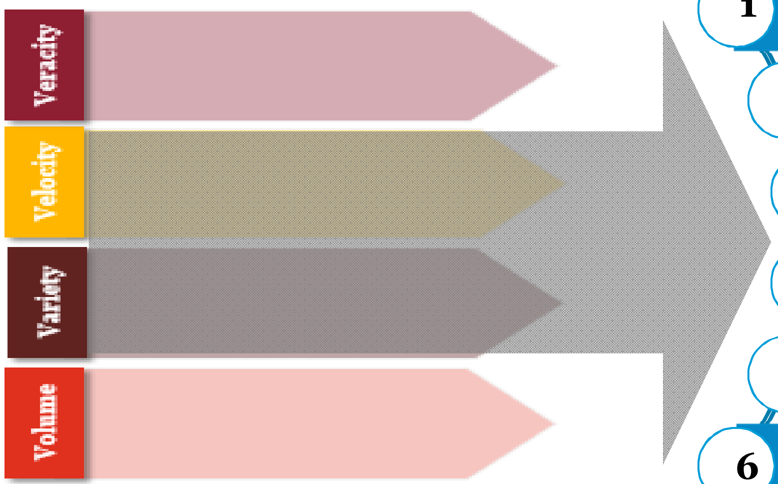
Graphic and definitions based on "Big Data in Action for Development," World Bank, worldbank.org

It’s not just about the data…

It is important to understand the distinction between Big Data sets (large, unstructured, fast, and uncertain data) and ‘Big Data Analytics’.

BigData + BigData Analytics

BigData — Refers to the DATA only

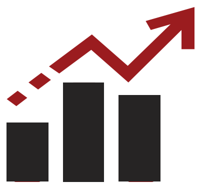
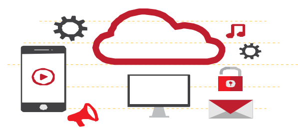


BigData Analytics — **Methods** of using Big Data to generate insight

Method	Description
1 Machine Learning/Deep Learning	Leveraging a computer’s ability to learn without being explicitly programmed to solve business problems
2 IoT (Internet of Things) & Sensor Analytics	Understanding value drivers from the ever-growing network of connected physical objects and the communication between them
3 Modeling Willingness-to-Pay	Mining product reviews to estimate willingness-to-pay for product features
4 Natural Language Processing	Understanding human speech as it is spoken through application of computer science, AI, and computational linguistics
5 Analyzing Data @ Scale	Using distributed computing and machine learning tools to analyze hundreds of gigabytes of data
6 Creating a Streaming Consumer Behavior Data Lake	Mining social data in real time to understand when and where consumers are making choices

… It's also about what, how, and why you use it

BigData Analytics – the process of harnessing Big Data to yield actionable insights – is a combination of five key elements:

Decisions	*Analytics*	*Data*	*Technology*	*Mindset & Skills*
The value of Big Data Analytics is driven by the unique decisions facing leaders, companies, and countries today. In turn, the type, frequency, speed, and complexity of decisions drive how Big Data Analytics is deployed.	To leverage the variety and volume of Big Data while managing its volatility, advanced analytical approaches are necessary, such as natural language processing, network analysis, simulative modeling, artificial intelligence, etc.	Big Data Analytics is about operationalizing new and more data, but it is also about data quality, data interoperability, data disaggregation, and the ability to modularize data structures to quickly absorb new data and new types of data.	To store, manage, and use Big Data often requires investments in new technologies and data processing methods, such as distributed processing (e.g., Hadoop), NoSQL storage, and Cloud computing.	Big Data Analytics requires firm commitment to using analytics in decision-making; a decisive mentality capable of employing in-the-moment intelligence; and investment in analytical technology, resources, and skills.
				

BigData Analytical Capabilities

Continuing increases in processing capacity have opened the door to a range of advanced algorithms and modeling techniques that can produce valuable insights from Big Data.

Structured ↔ Unstructured

Traditional ↔ Emerging

Regression *Discover relationships between variables*	***Time Series Analysis*** *Discover relationships over time*	***Signal Analysis*** *Distinguish between noise and meaningful information*	***Cluster Analysis*** *Discover meaningful groupings of data points*
A/B/N Testing *Experiment to find the most effective variation of a website, product, etc*	***Classification*** *Organize data points into known categories*	***Simulation Modeling*** *Experiment with a system virtually*	***Spatial Analysis*** *Extract geographic or topological information*
Visualization *Use visual representations of data to find and communicate info*	***Predictive Modeling*** *Use data to forecast or infer behavior*	***Complex Event Processing*** *Combine data sources to recognize events*	***Sentiment Analysis*** *Extract consumer reactions based on social media behavior*
Network Analysis *Discover meaningful nodes and relationships on networks*	***Optimization*** *Improve a process or function based on criteria*	***Deep QA*** *Find answers to human questions using artificial intelligence*	***Natural Language Processing*** *Extract meaning from human speech or writing*

Forward-Looking vs. Rear-View Analytics

Big Data Analytics improves the speed and efficiency with which we understand the past, and opens up entirely new avenues for preparing for and adapting to the future.

Rear-view | **Forward-looking**

Increasing Business Value

Descriptive Analytics

What happened?
Describe, summarize and analyze historical data

- ÅObserved behavior or events
- ÅNon-traditional data sources such as social listening and web crawling

Diagnostic Analytics

Why did it happen?
Identify causes of trends and outcomes

- ÅObserved behavior or events
- ÅNon-traditional data sources such as social listening and web crawling
- ÅStatistical and regression analysis
- ÅDynamic visualization

Predictive Analytics

What could happen?
Predict future outcomes based on the past

- ÅForward-looking view of current and future value
- ÅSentiment Scoring
- ÅGraph analysis and Natural Language Processing to identify hidden relationships and themes
- ÅDual objective models
- ÅBehavioral economics

Prescriptive Analytics

What should be done?
Recommend 'right' or optimal actions or decisions

- ÅReal-time product and service propositions (graph analysis, entity resolution on data lakes to infer present customer need)
- ÅRapid evaluation of multiple 'what-if' scenarios
- ÅOptimization decisions and actions

Continuous Analytics

How do we adapt to change?
Monitor, decide, and act autonomously or semi-autonomously

- ÅMonitor results on a continuous basis
- ÅDynamically adjust strategies based on changing environment and improved predictions
- ÅAgent-based and dynamic simulation models, time-series analysis

Increasing Sophistication of Data & Analytics

MAPREDUCE

THE BIRTH OF BIGDATA TECHNOLOGY

MapReduce

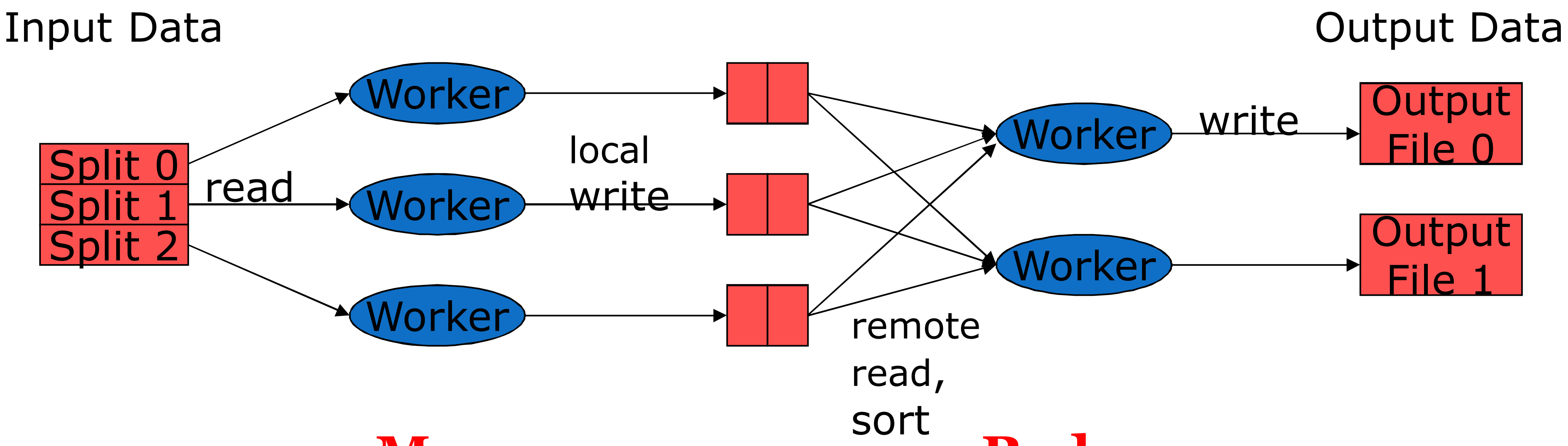
- MapReduce [OSDI'04] provides
  - Automatic parallelization, distribution
  - I/O scheduling
    - Load balancing
    - Network and data transfer optimization
  - Fault tolerance
    - Handling of machine failures
- **Need more power: Scale out, not up!**
    - Large number of **commodity servers** as opposed to some high end specialized servers

Apache Hadoop:
Open source implementation of MapReduce

Typical problem solved by MapReduce

- Read a lot of data
- Map: extract something you care about from each record
- Shuffle and Sort
- Reduce: aggregate, summarize, filter, or transform
- Write the results

MapReduce workflow



Map

extract something you care about from each record

Reduce

aggregate, summarize, filter, or transform

Mappers and Reducers

- Need to handle more data? Just add more Mappers/Reducers!
- No need to handle multithreaded code ☺
  - Mappers and Reducers are typically single threaded and deterministic
    - Determinism allows for restarting of failed jobs
  - Mappers/Reducers run entirely independent of each other
    - In Hadoop, they run in separate JVMs

Example: Word Count

http://kickstarthadoop.blogspot.ca/2011/04/word-count-hadoop-map-reduce-example.html

Mapper

- Reads in input pair <Key,Value>
- Outputs a pair <K', V'>
  - Let's count number of each word in user queries (or Tweets/Blogs)
  - The input to the mapper will be <queryID, QueryText>:

```
<Q1,"The teacher went to the store. The store was
closed; the store opens in the morning. The store opens
at 9am." >
```

  - The output would be:

```
<The, 1> <teacher, 1> <went, 1> <to, 1> <the, 1>
<store,1> <the, 1> <store, 1> <was, 1> <closed, 1>
<the, 1> <store,1> <opens, 1> <in, 1> <the, 1>
<morning, 1> <the 1> <store, 1> <opens, 1> <at, 1>
<9am, 1>
```

Reducer

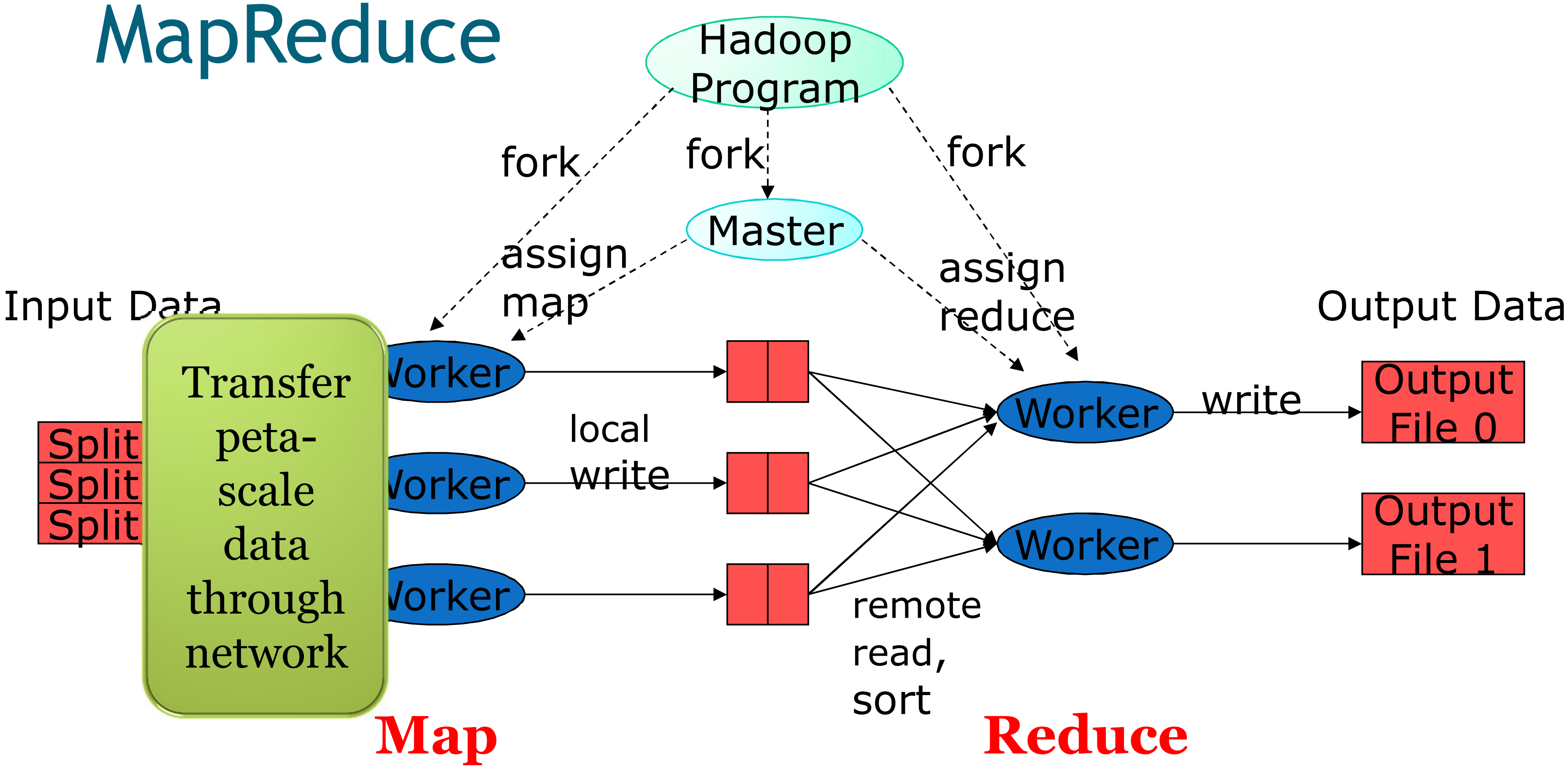
- Accepts the Mapper output, and aggregates values on the key
  - For our example, the reducer input would be:

    <The, 1> <teacher, 1> <went, 1> <to, 1> <the, 1> <**store**, 1>
    <the, 1> <store, 1> <was, 1> <closed, 1> <the, 1> <**store**, 1>
    <opens,1> <in, 1> <the, 1> <morning, 1> <the 1> <**store**, 1>
    <opens, 1> <at, 1> <9am, 1>

  - The output would be:

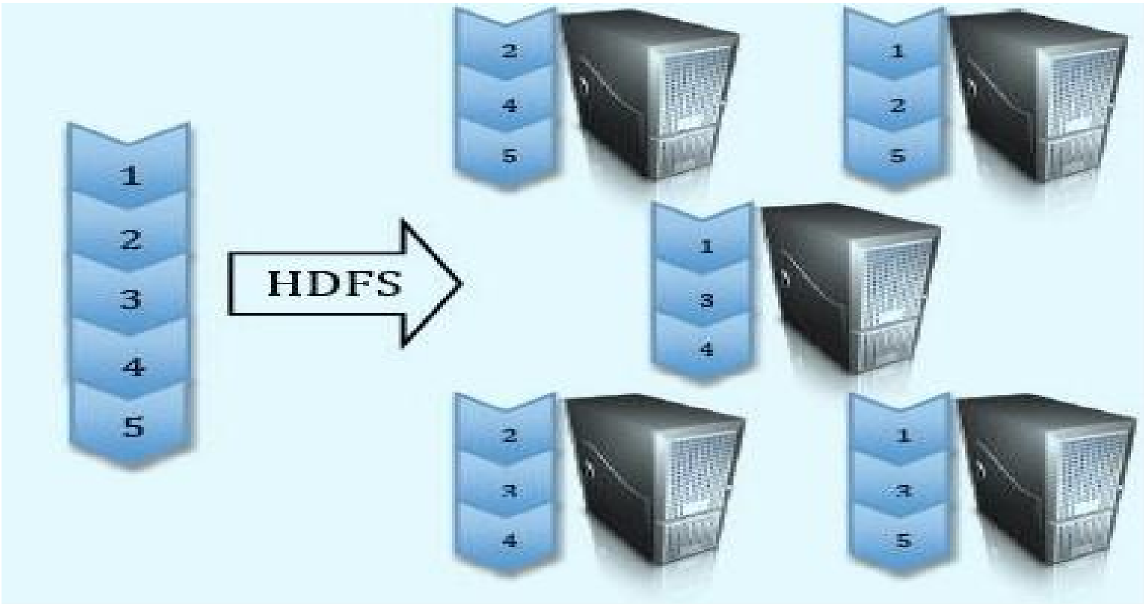
    <The, 6> <teacher, 1> <went, 1> <to, 1> **<store, 3>** <was, 1>
    <closed, 1> <opens, 1> <morning, 1> <at, 1> <9am, 1>

MapReduce

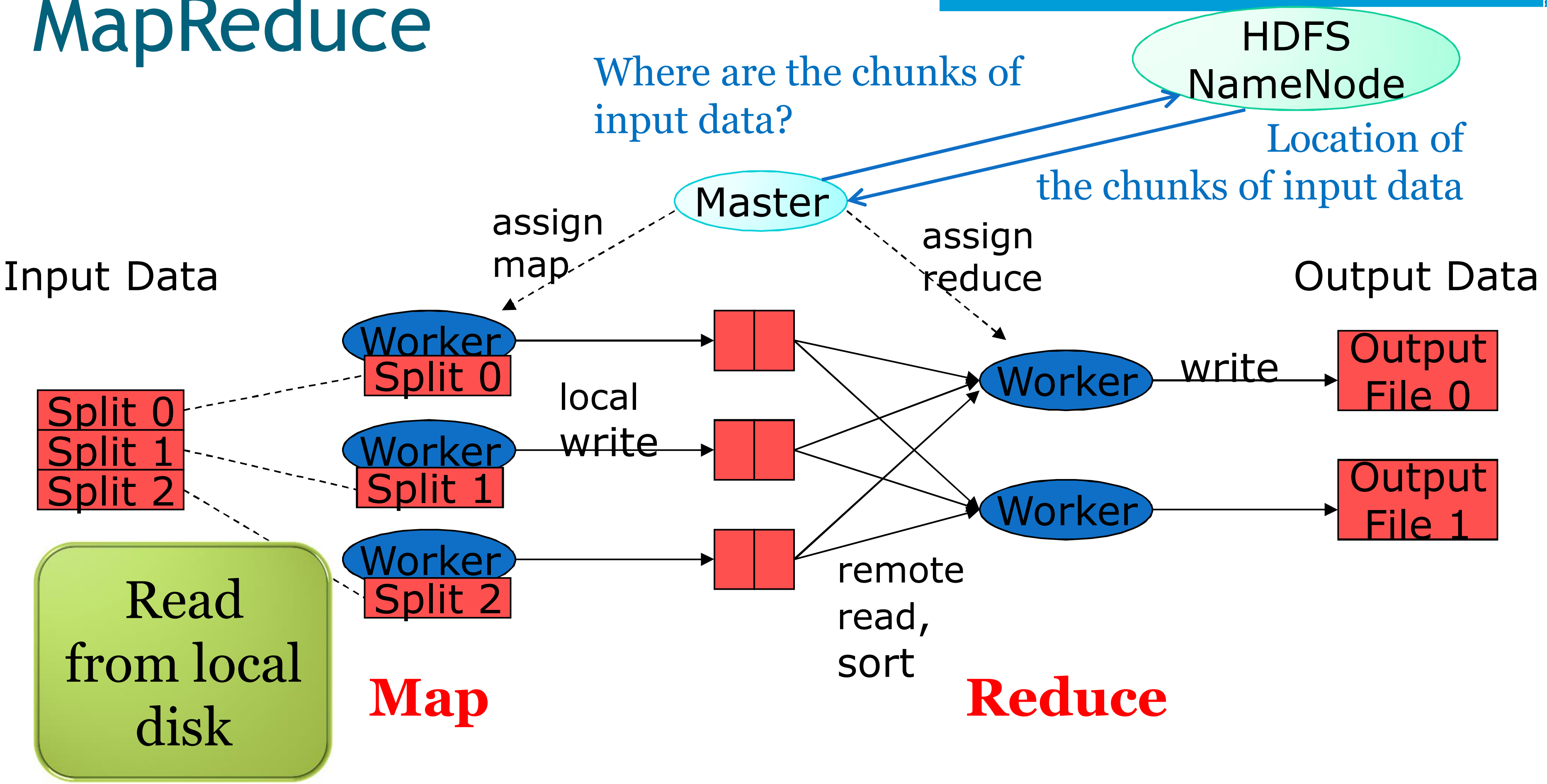

Hadoop Program
fork
fork
fork
Master
assign map
assign reduce
Input Data
Split
Split
Split
Transfer peta-scale data through network
Worker
Worker
Worker
local write
remote read, sort
Worker
Worker
write
Output Data
Output File 0
Output File 1
Map
Reduce

Google File System (GFS)
Hadoop Distributed File System (HDFS)

- Split data and store 3 replica on commodity servers



MapReduce


Where are the chunks of input data?
HDFS NameNode
Location of the chunks of input data
Master
assign map
assign reduce
Input Data
Output Data
Worker
Split 0
Split 0
Split 1
Split 2
Worker
Split 1
Worker
Split 2
local write
Worker
write
Output File 0
Worker
Output File 1
remote read, sort
Read from local disk
Map
Reduce

Locality Optimization

- **Master scheduling policy:**
  - Asks GFS for locations of replicas of input file blocks
  - Map tasks scheduled so GFS input block replica are on same machine or same rack
- Effect: Thousands of machines read input at local disk speed
  - Eliminate network bottleneck!

Failure in MapReduce

- Failures are norm in commodity hardware
- **Worker** failure
  - Detect failure via periodic heartbeats
  - Re-execute in-progress map/reduce tasks
- **Master** failure
  - Single point of failure; Resume from Execution Log
- **Robust**
  - Google's experience: lost 1600 of 1800 machines once!, but finished fine

Fault tolerance: Handled via re-execution

- On worker failure:
  - Detect failure via periodic heartbeats
  - Re-execute completed and in-progress *map* tasks
  - Task completion committed through master
- Robust: [Google's experience] lost 1600 of 1800 machines, but finished fine

Refinement: Redundant Execution

- **Slow workers** significantly lengthen completion time
  - Other jobs consuming resources on machine
  - Bad disks with soft errors transfer data very slowly
  - Weird things: processor caches disabled (!!)
- **Solution**: spawn backup copies of tasks
  - Whichever one finishes first "wins"

Refinement: Skipping Bad Records

Map/Reduce functions sometimes fail for particular inputs

- Best solution is to debug & fix, but not always possible
- If master sees two failures for the same record:
  - Next worker is told to skip the record

Summary

- MapReduce
  - Programming paradigm for data-intensive computing
  - Distributed & parallel execution model
  - Simple to program
    - The framework automates many tedious tasks (machine selection, failure handling, etc.)

Details of GFS

Motivation: Large Scale Data Storage

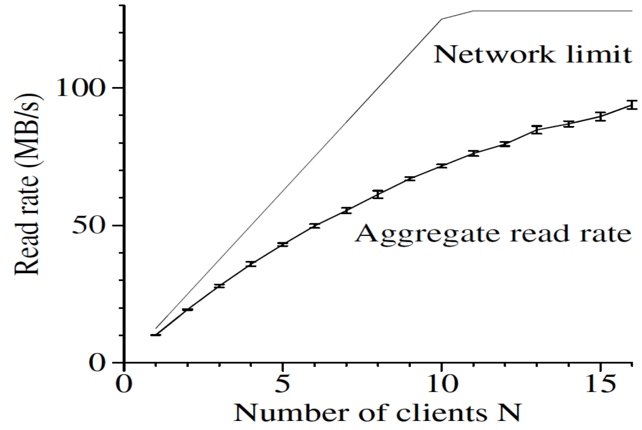
- Manipulate large (Peta Scale) sets of data
- Large number of machines with commodity hardware
- Component failure is the norm

- Goal: **Scalable**, **high performance**, **fault tolerant** distributed file system

Why a new file system?

- None designed for their failure model
- Few scale as highly or dynamically and easily
- Lack of special primitives for large distributed computation

What should expect from GFS



- Designed for Google's application
  - Control of both file system and application
  - Applications use a few specific access patterns
    - Append to larges files
    - Large streaming reads
  - Not a good fit for
    - low-latency data access
    - lots of small files, multiple writers, arbitrary file modifications
- Not POSIX, although mostly traditional
  - Specific operations: RecordAppend

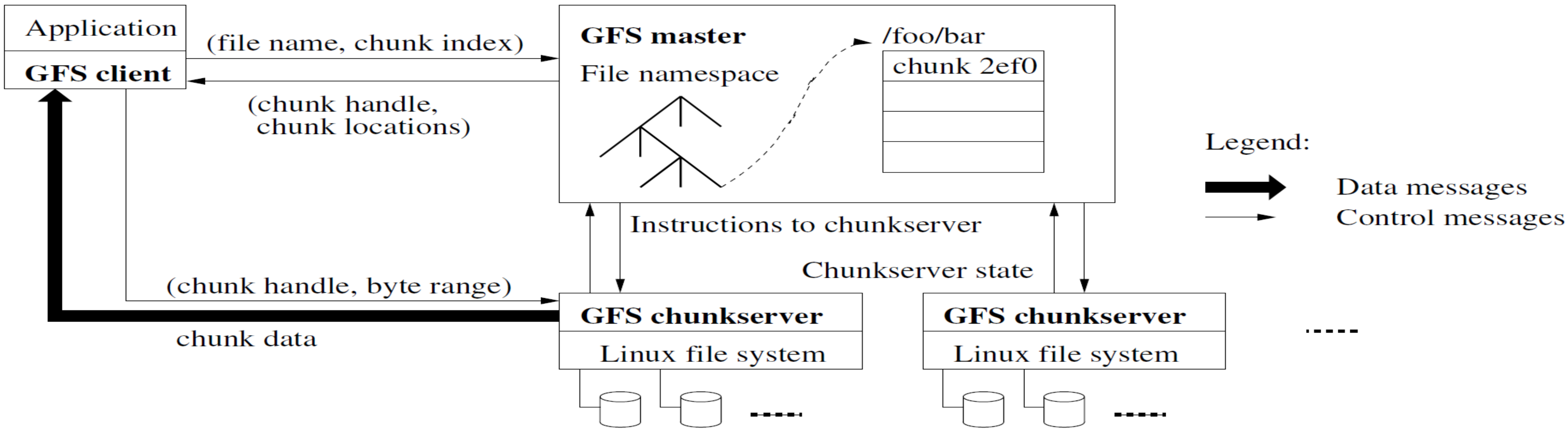
- Different characteristic than transactional or the "customer order" data : "write once read many (WORM)"
  - e.g. web logs, web crawler's data, or healthcare and patient information
  - WORM inspired MapReduce programming model
- Google exploited this characteristics in its Google file system [SOSP'03]
  - Apache Hadoop: Open source project

Components

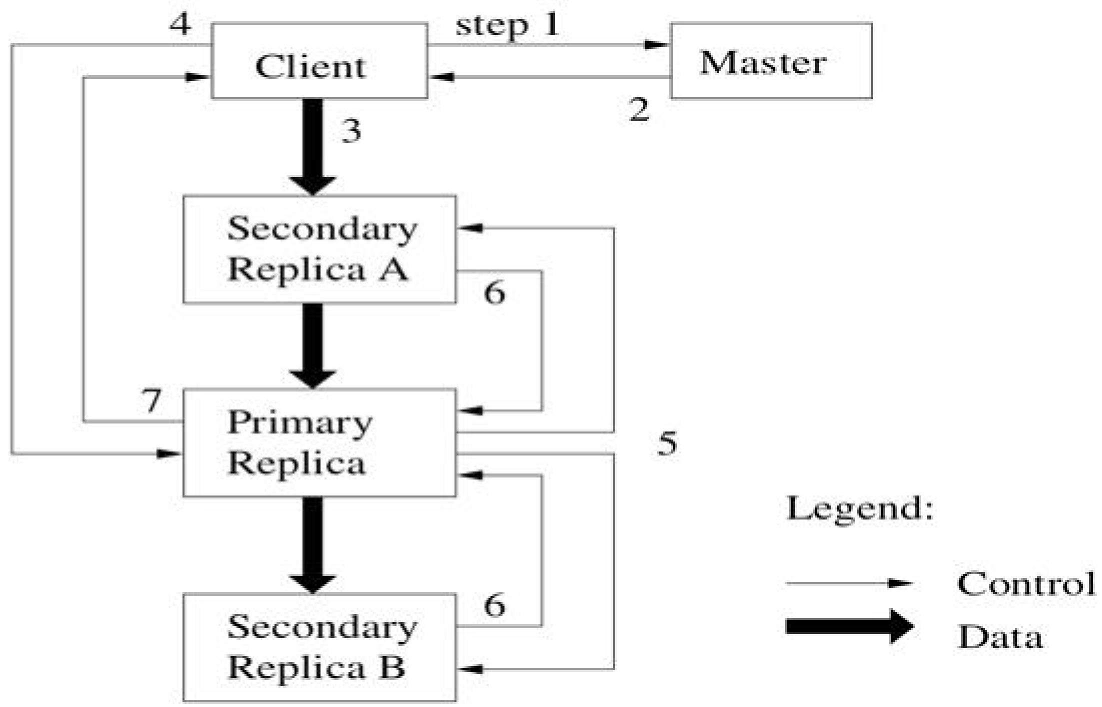
- **Master (NameNode)**
  - Manages metadata (namespace)
  - Not involved in data transfer
  - Controls allocation, placement, replication
- **Chunkserver (DataNode)**
  - Stores chunks of data
  - No knowledge of GFS file system structure
  - Built on local Linux file system

www.cse.buffalo.edu/~okennedy/courses/cse704fa2012/2.2-HDFS.pptx

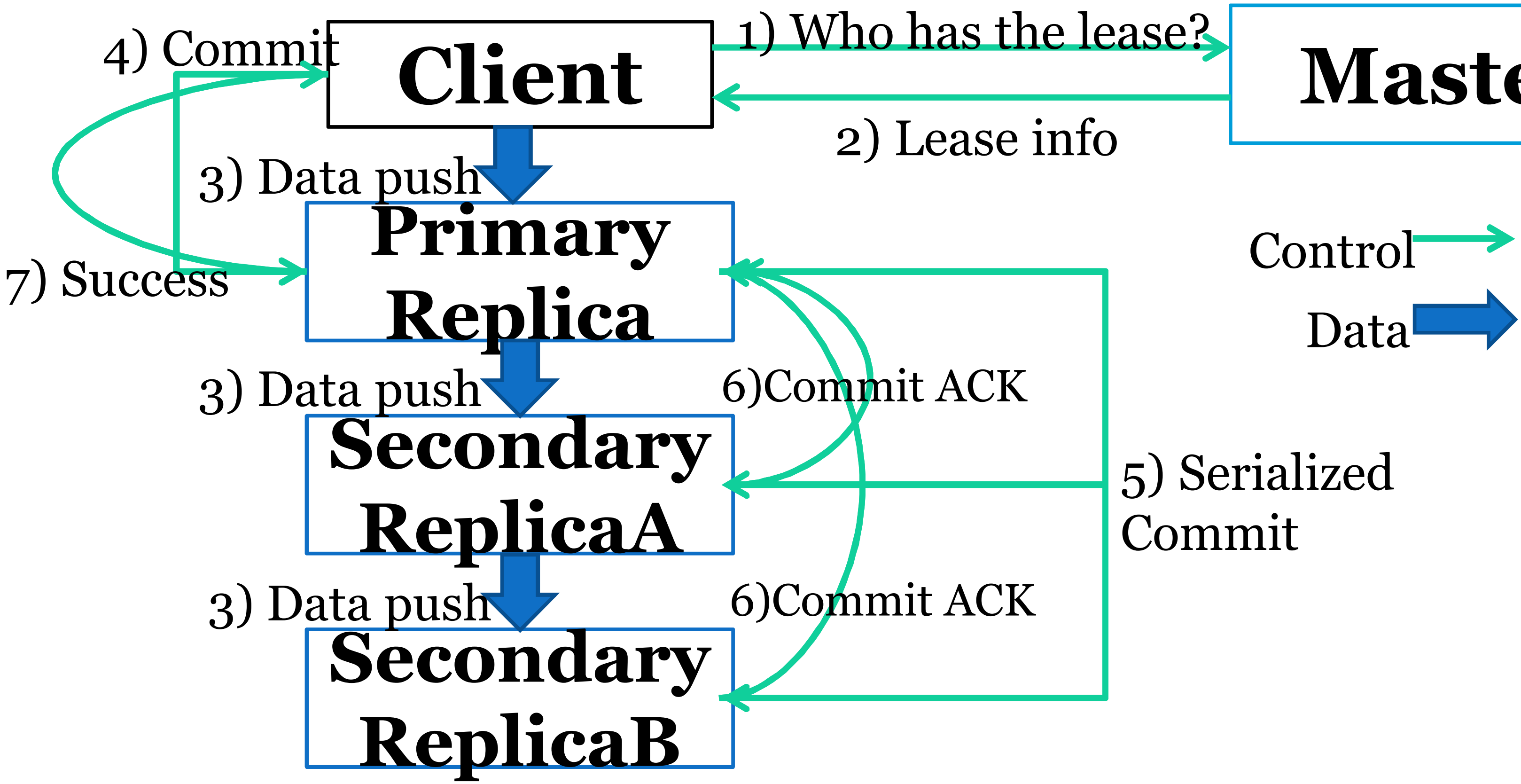
GFS Architecture



Write operation



Write(filename, offset, data)



RecordAppend(filename, data)

- Significant use in distributed apps. For example at Google production cluster:
  - 21% of bytes written
  - 28% of write operations
- Guaranteed: All data appended at least once as a single consecutive byte range
- Same basic structure as write
  - Client obtains information from master
  - Client sends data to data nodes (chunkservers)
  - Client sends "append-commit"
  - Lease holder serializes append
- **Advantage:** Large number of concurrent writers with minimal coordination

RecordAppend (2)

- Record size is limited by chunk size
- When a record does not fit into available space,
  - chunk is padded to end
  - and client retries request.

Fault tolerance

- Replication
  - High availability for reads
  - User controllable, default 3 (non-RAID)
  - Provides read/seek bandwidth
  - Master is responsible for directing re-replication if a data node dies
- Online checksumming in data nodes
  - Verified on reads

Replica Management

- Bias towards topological spreading
  - Rack, data center
- Rebalancing
  - Move chunks around to balance disk fullness
  - Gently fixes imbalances due to:
    - Adding/removing data nodes

Replica Management (Cloning)

- Chunk replica lost or corrupt
- Goal: minimize app disruption and data loss
  - Approximately in priority order
    - More replica missing-> priority boost
    - Deleted file-> priority decrease
    - Client blocking on a write-> large priority boost
  - Master directs copying of data
- Performance on a production cluster
  - Single failure, full recovery (600GB): 23.2 min
  - Double failure, restored 2x replication: 2min

Garbage Collection

- Master does **not** need to have a strong knowledge of what is stored on each data node
  - Master regularly scans namespace
  - After GC interval, deleted files are removed from the namespace
  - Data node periodically polls Master about each chunk it knows of.
  - If a chunk is forgotten, the master tells data node to delete it.

Limitations

- Master is a central point of failure
- Master can be a scalability bottleneck
- Latency when opening/stating thousands of files
- Security model is weak

Conclusion

- Inexpensive commodity components can be the basis of a large scale reliable system
- Adjusting the API, e.g. RecordAppend, can enable large distributed apps
- Fault tolerant
- Useful for many similar apps