CMS/CS/EE 144

Networks: Structure & Economics

Administrivia

1) HW4 is due thursday
2) Don't forget your blog posts!
3) Start thinking about projects…
4) Rankmaniac is coming up…
   → Don't forget to email the Tas with your teams!

COURSE OUTLINE

1) Understanding network structure
2) <u>Exploiting network structure</u>
3) Network economics
4) Data center design

Where can network structure be exploited?

TODAY

Search … *the killer app for network structure*

High clustering both aids and inhibits spread of viruses (or information)

Heavy-tailed degrees make networks "robust yet fragile" to attack

Identifying "fake" accounts and suggesting friends in facebook

…many other places…

<u>Search engine goal:</u> Make money

<u>Challenges:</u>

-- Need lots of users

-- Need to serve ads effectively

Later in course

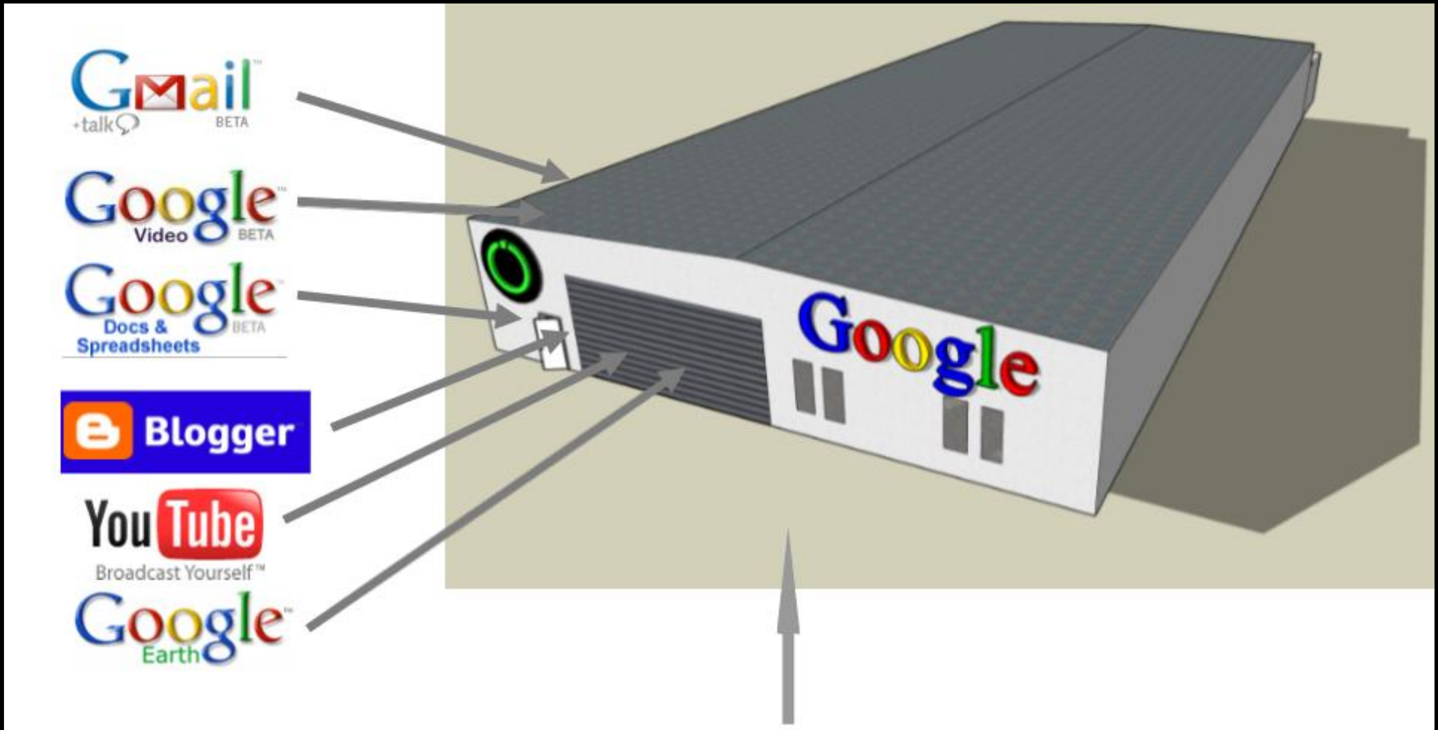
Give users what
they want (speed & quality)

Data centers!

Last time

What are data centers?

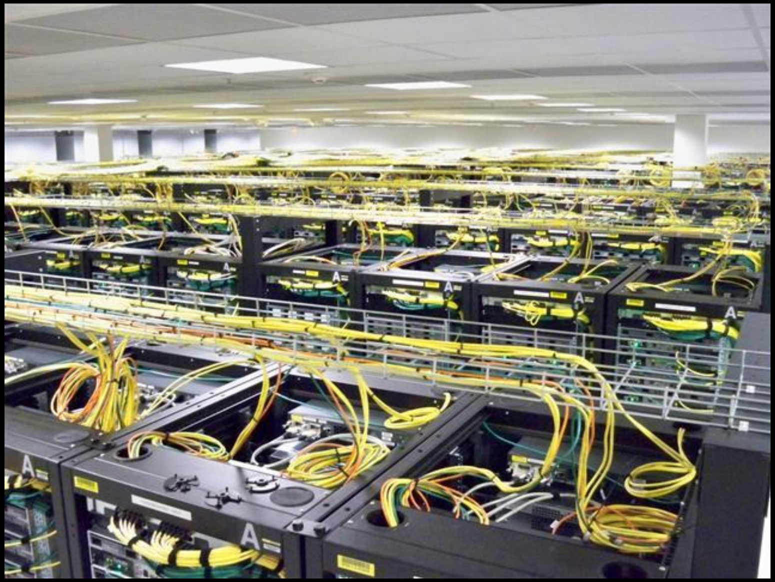
"A large group of networked computer servers typically used by organizations for the remote storage, processing, or distribution of large amounts of data."



A "warehouse-scale" computer

What are data centers?

My answer → Information Factories

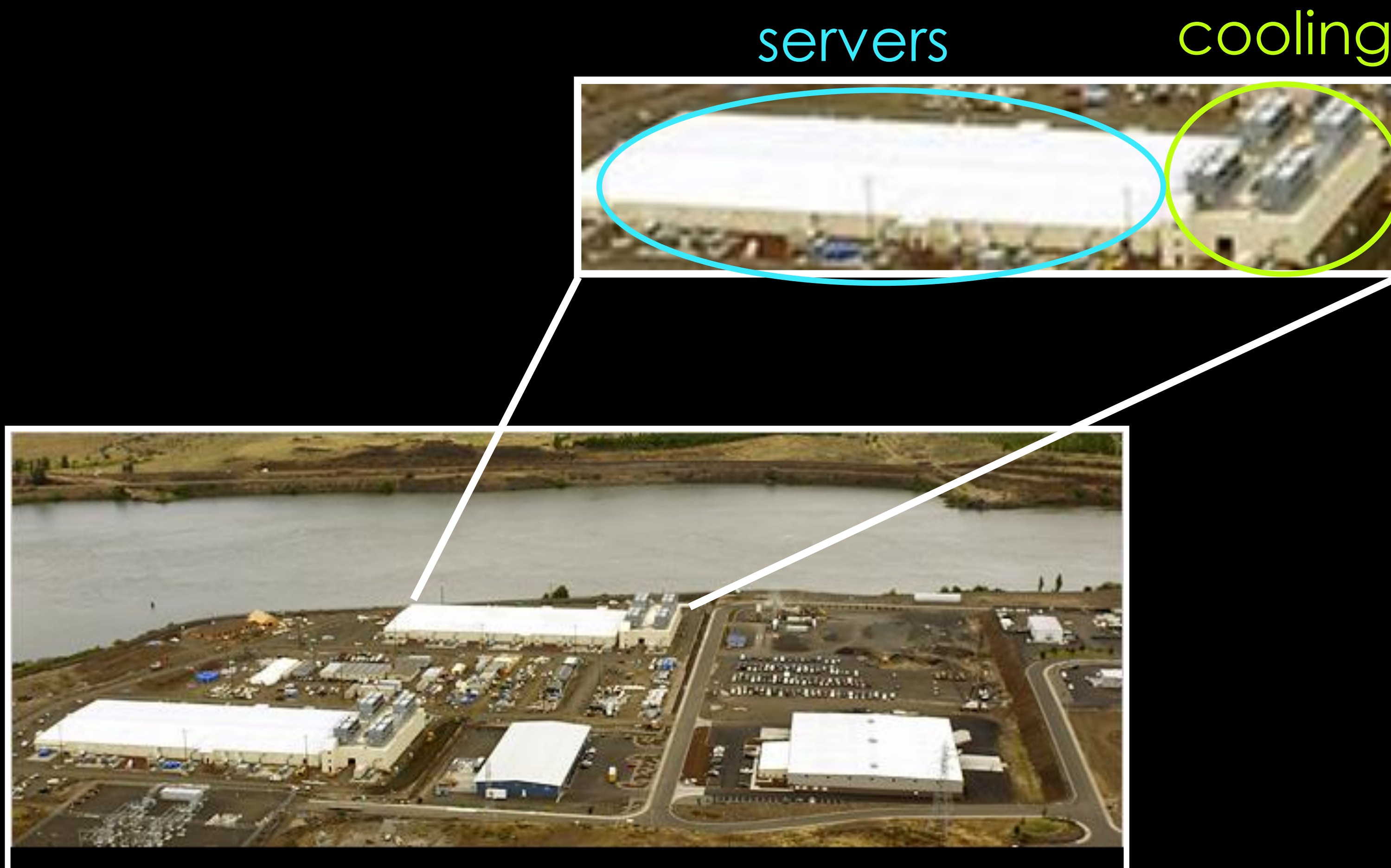


What are data centers?

My answer → <u>Information Factories</u>

What are data centers?

My answer → <u>Information Factories</u>



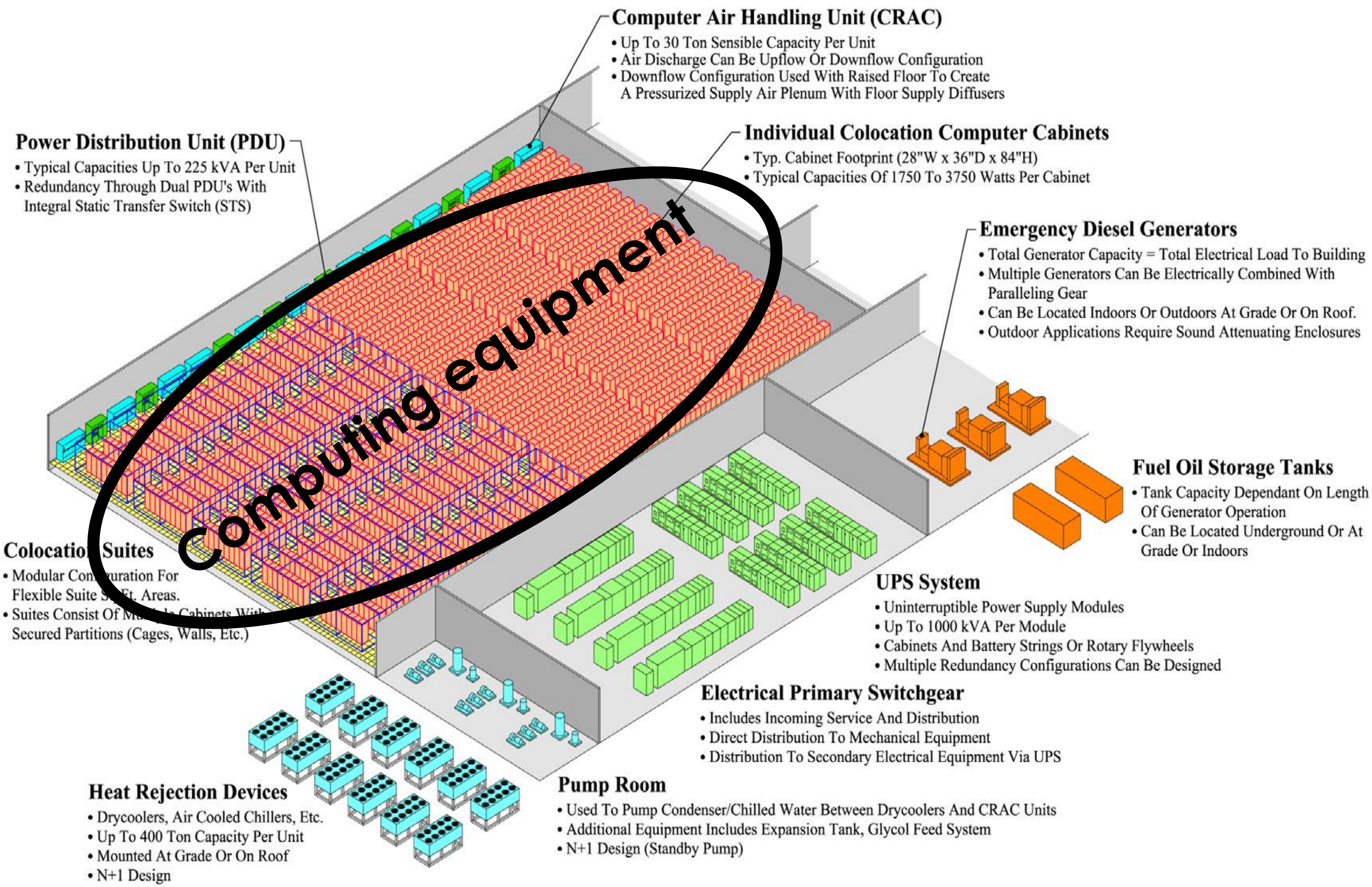


Image courtesy of DLB Associates

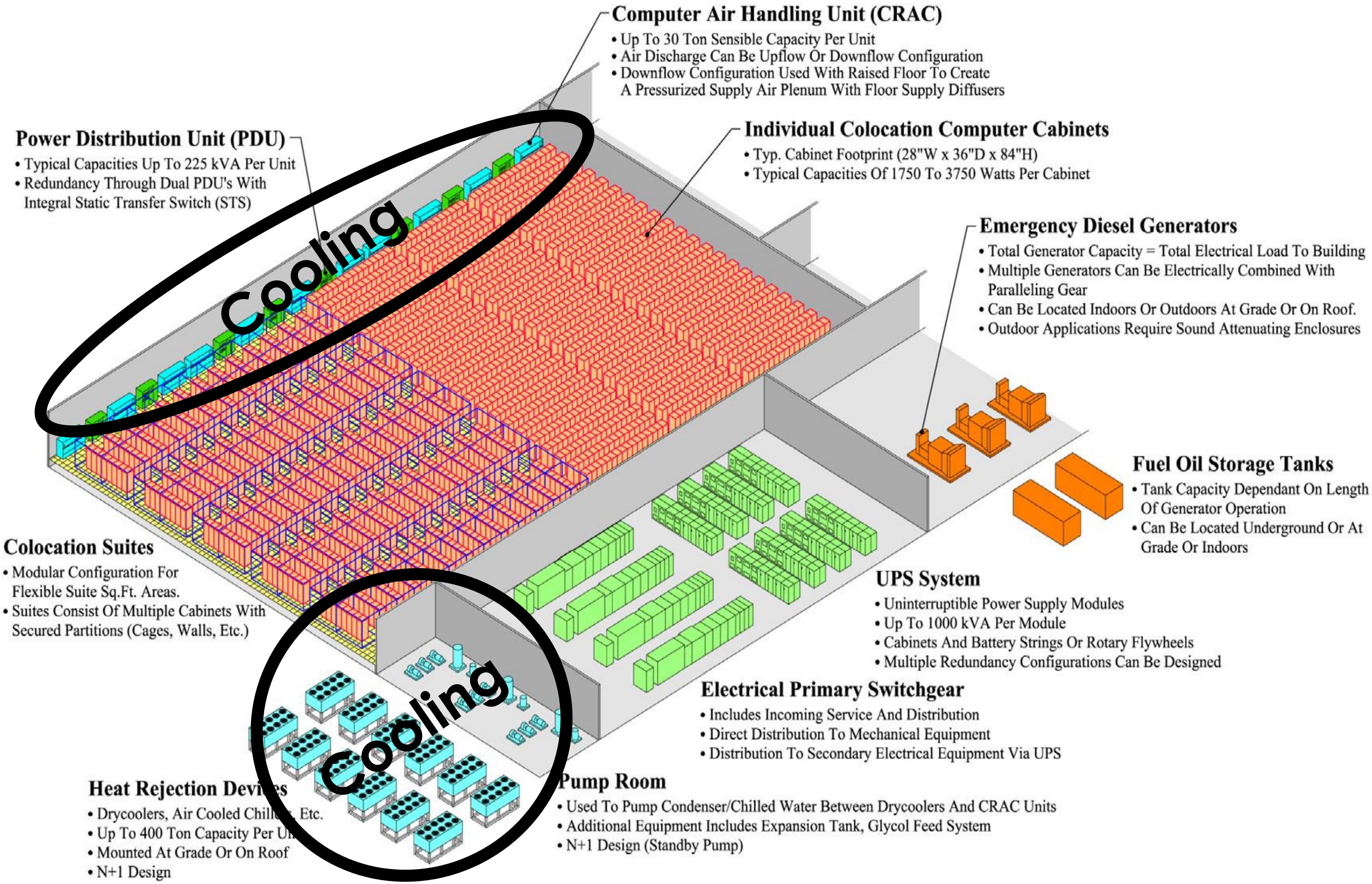
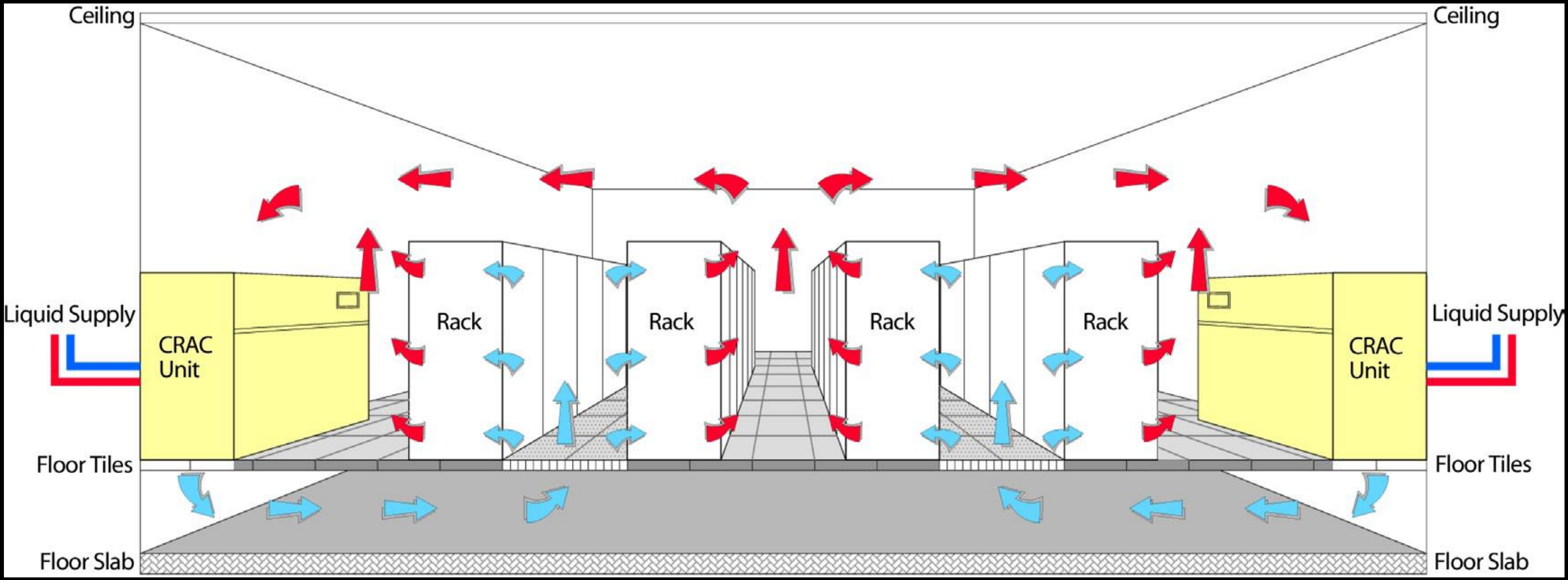


Image courtesy of DLB Associates



CRAC (Computer Room AC) – “standard” air conditioning

“Free” cooling – outside air used to cool the coolant for the AC (evaporative cooling)

Liquid cooling – Bring chilled water into the racks themselves

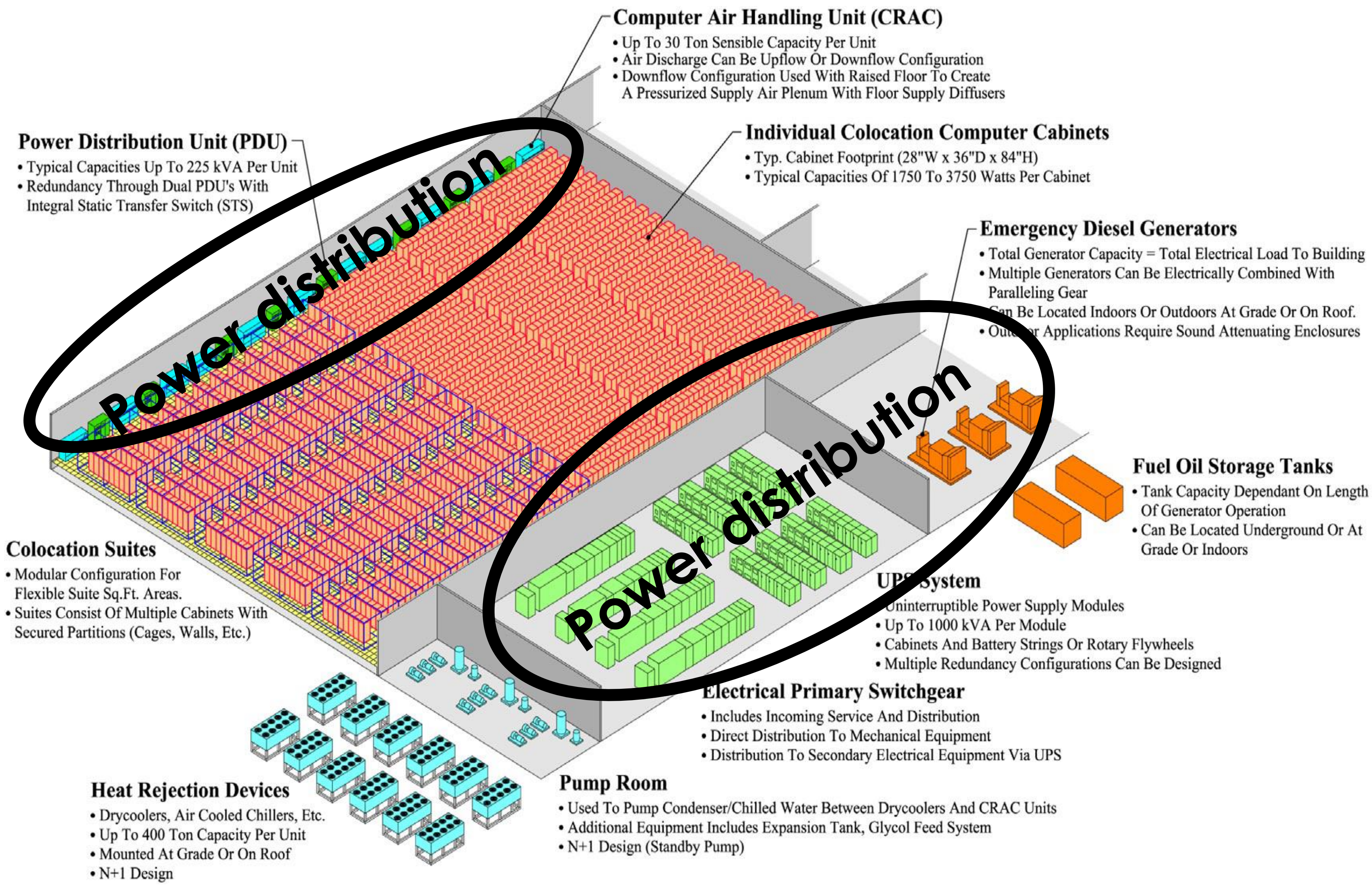
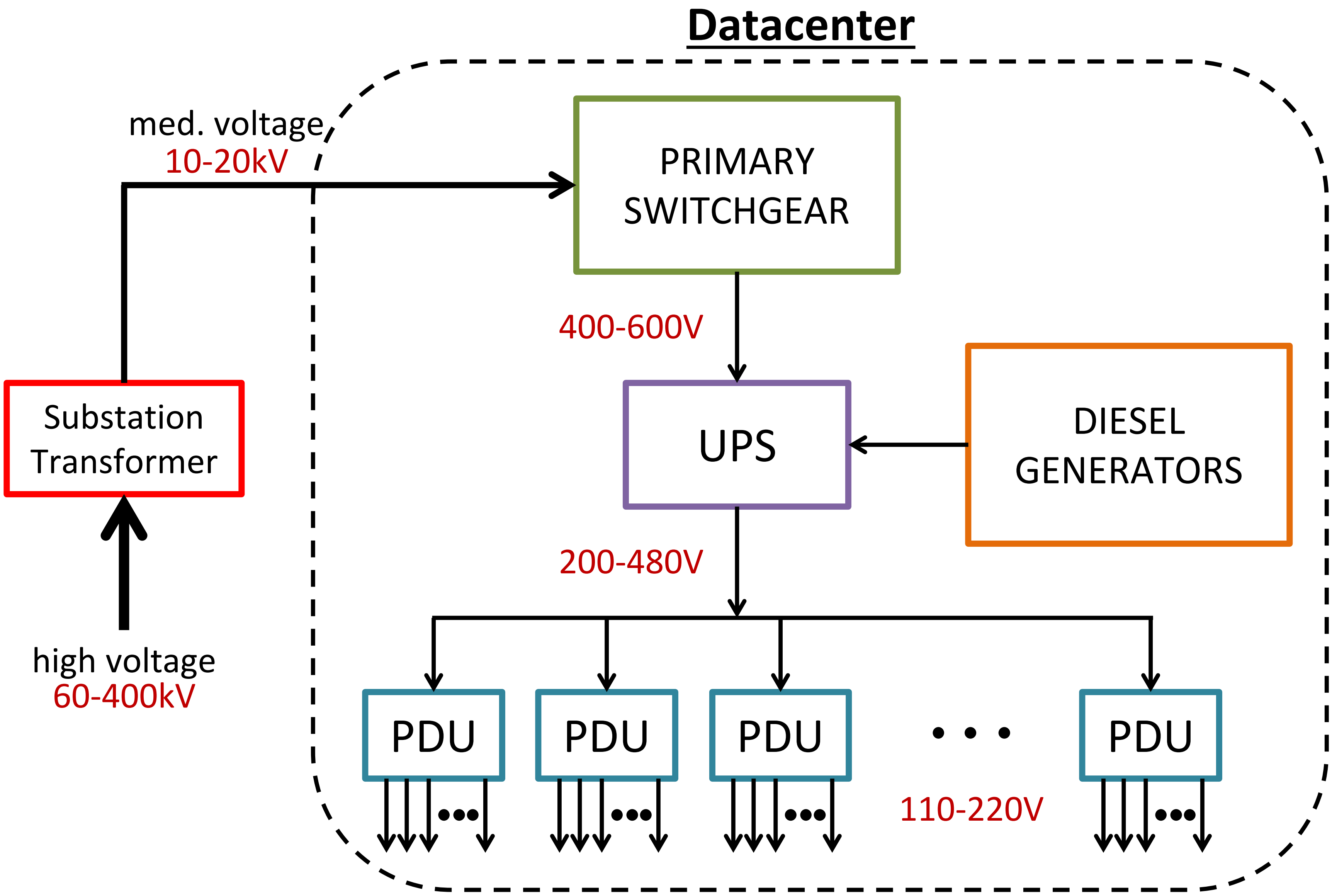


Image courtesy of DLB Associates


Datacenter
med. voltage
10-20kV
PRIMARY SWITCHGEAR
400-600V
UPS
DIESEL GENERATORS
200-480V
Substation Transformer
high voltage
60-400kV
PDU
PDU
PDU
PDU
110-220V

PRIMARY SWITCHGEAR

- Breakers for fault protection
- Transformers for voltage scaling

UPS

- Transfer switch for selecting power source
- Batteries/flywheels (AC-DC-AC double conversion)
- Conditions incoming power feed (removes spikes, distortions)

PDU

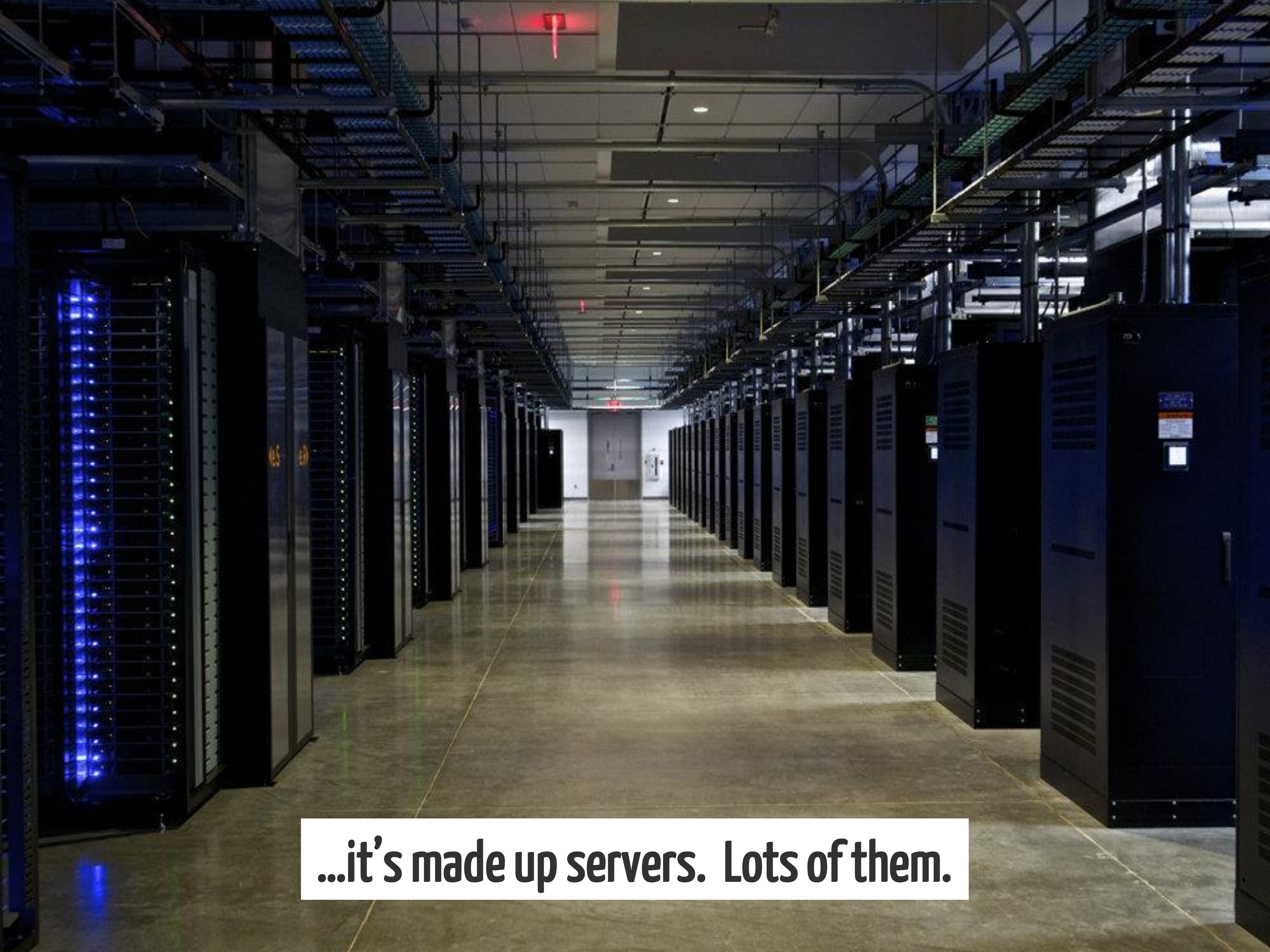
- Similar to breaker panels in residential houses
- Break a higher-voltage feed into several 110- or 220-V circuits that feed servers
- Typically handles 75-225kW of load (each circuit handles up to 6kW)
- Often provide additional redundancy

KEY ISSUES FOR DATA CENTERS

1. Energy usage
2. Ease of programming
3. Reliability
4. Resource allocation (Performance issues)

How clean is the cloud?

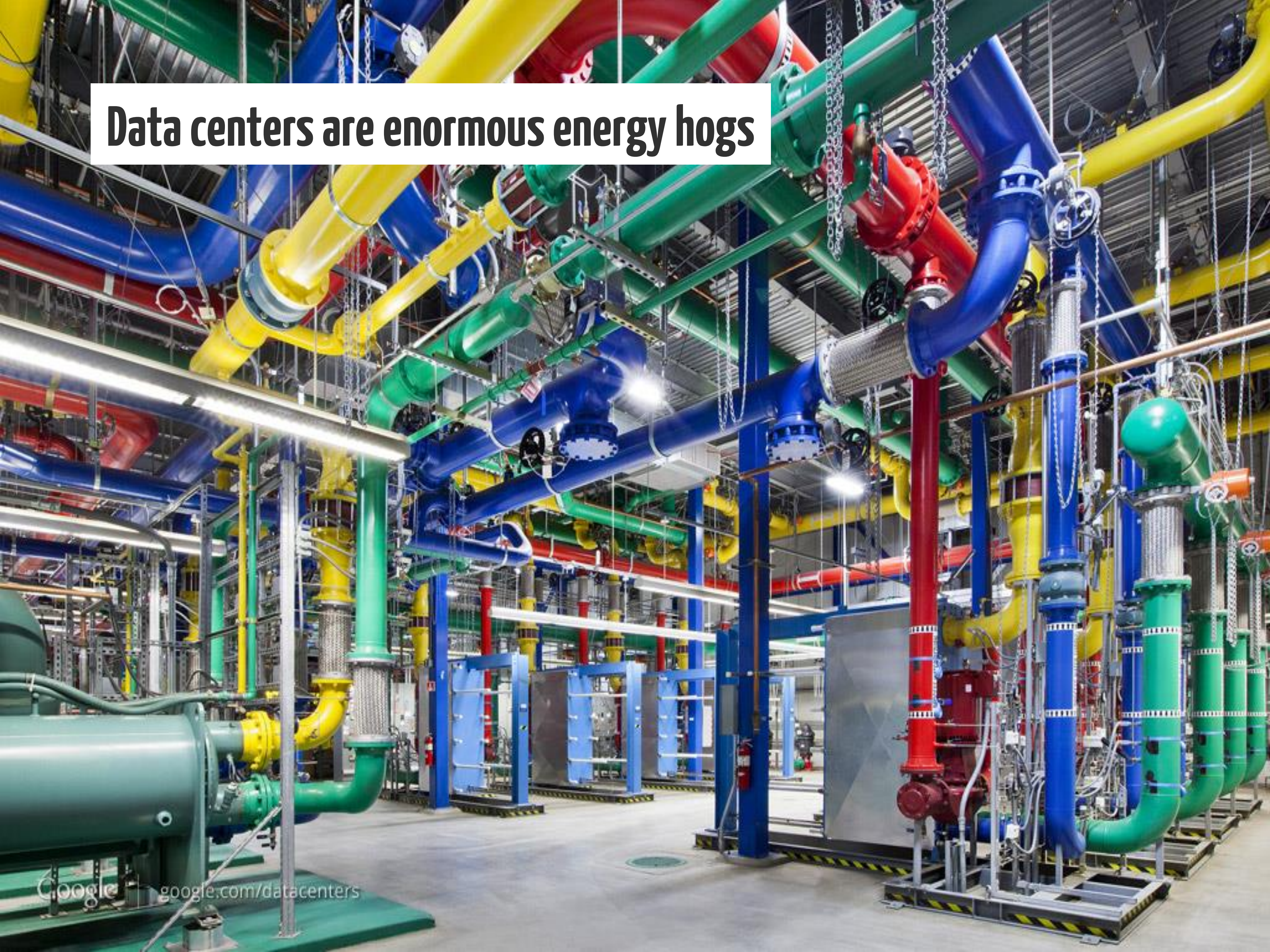
It sounds light & ephemeral, but...


...it’s made up servers. Lots of them.


Google alone spends $5+ billion/quarter on data centers!

Data centers are enormous energy hogs

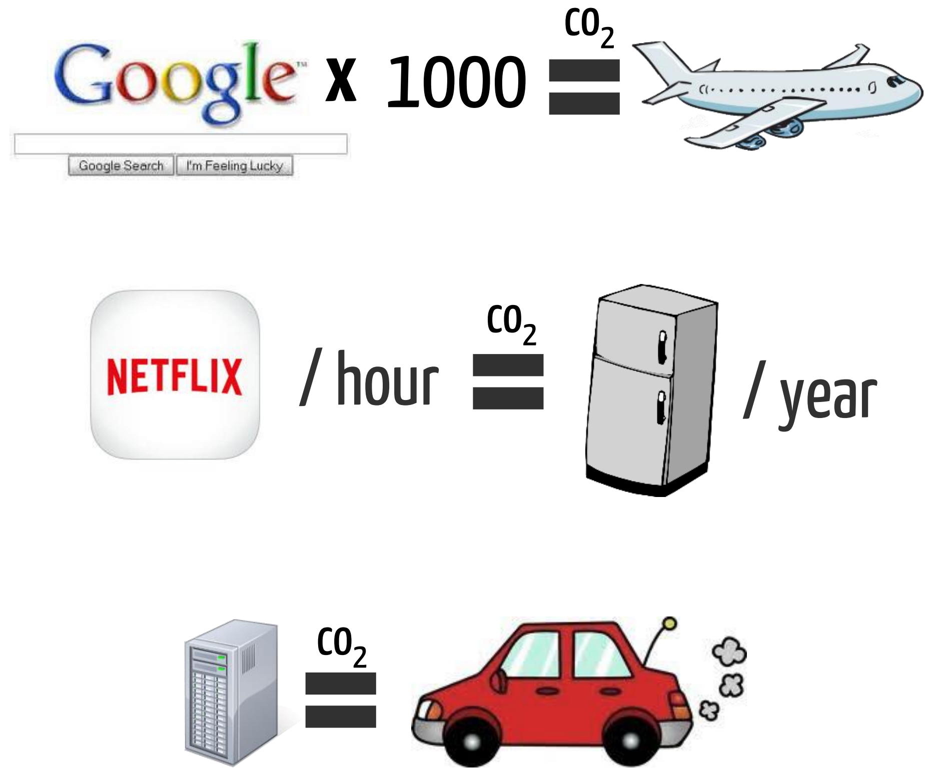
Data centers are enormous energy hogs



“We need to recognize that by the middle of the next decade data centers will rank among the largest users of electrical power on the planet.”

-- Brad Smith (President, Microsoft)

Data centers already create more emissions than the airline industry!

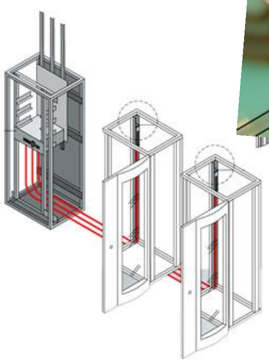


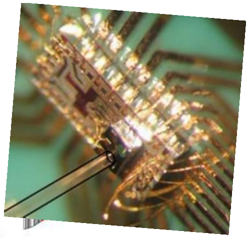
Google
Google Search
I'm Feeling Lucky
x 1000
CO2
NETFLIX
/ hour
CO2
/ year
CO2

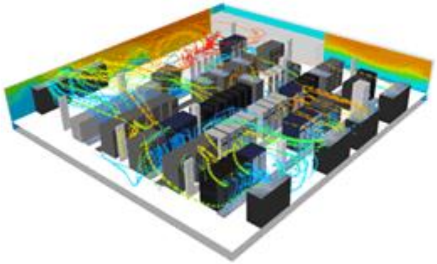
This has led to lots of attention in the press...

...and a huge push from academia & industry

Lots of multi-disciplinary research







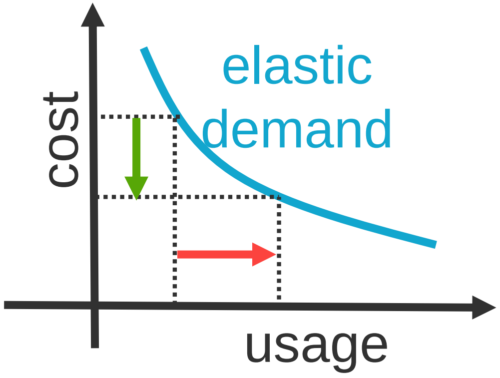
$$\text{Power Usage Effectiveness (PUE)} = \frac{\text{Power to building}}{\text{Power to computing}}$$

$$\overbrace{\underbrace{>4 \longrightarrow <1.1}}$$

lower PUE ≠ lower energy consumption

& never forget Jevons' paradox!

“It is wholly a confusion of ideas to suppose that the economical use of fuel is equivalent to a diminished consumption. The very contrary is the truth.”



-- William Stanley Jevons, 1865

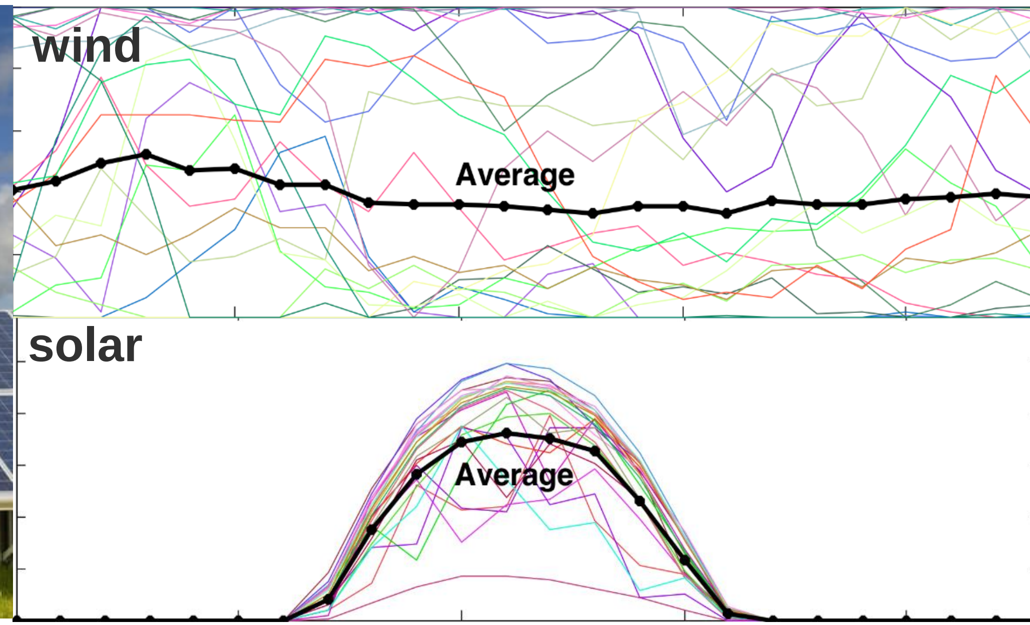
$$\text{Power Usage Effectiveness (PUE)} = \frac{\text{Power to building}}{\text{Power to computing}}$$

$>4 \longrightarrow <1.1$

lower PUE ≠ lower energy consumption

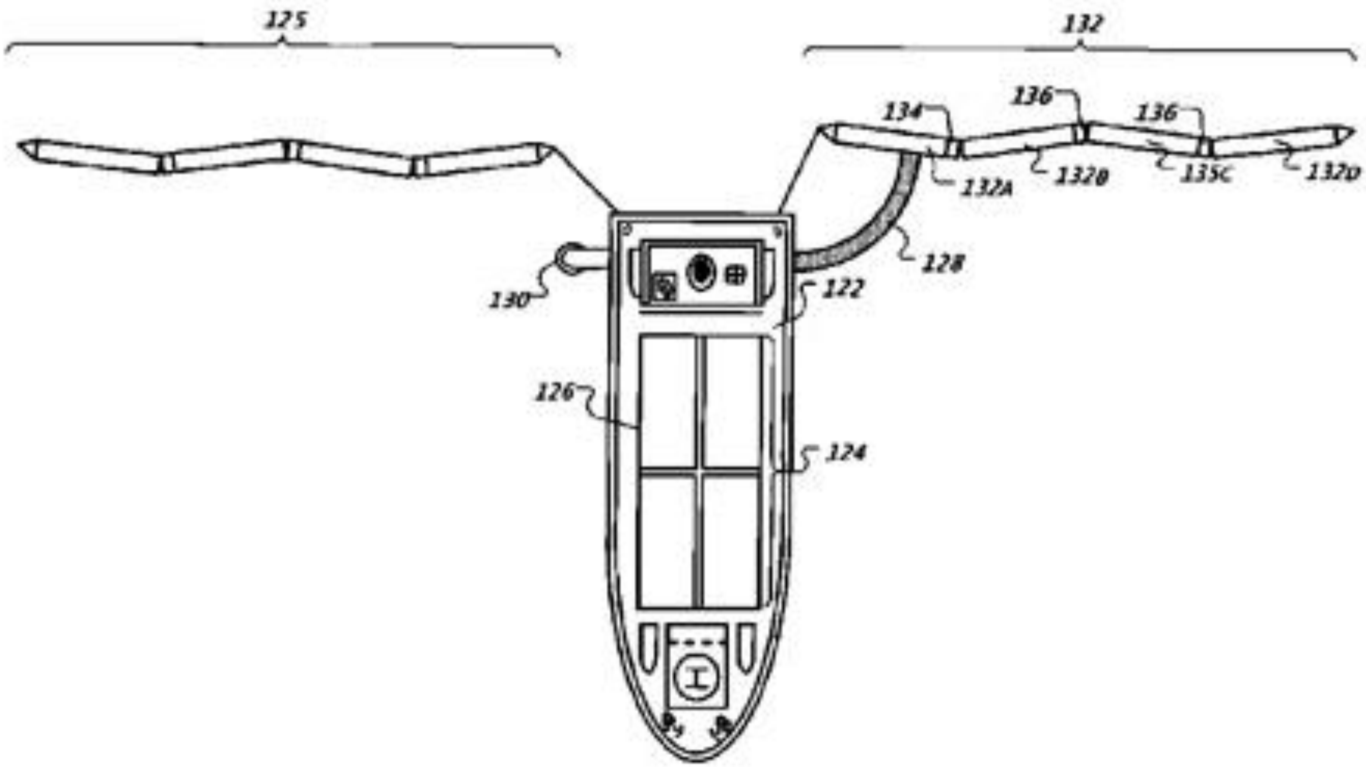
Sustainable data centers

Can a data center run (almost) entirely on renewable sources?



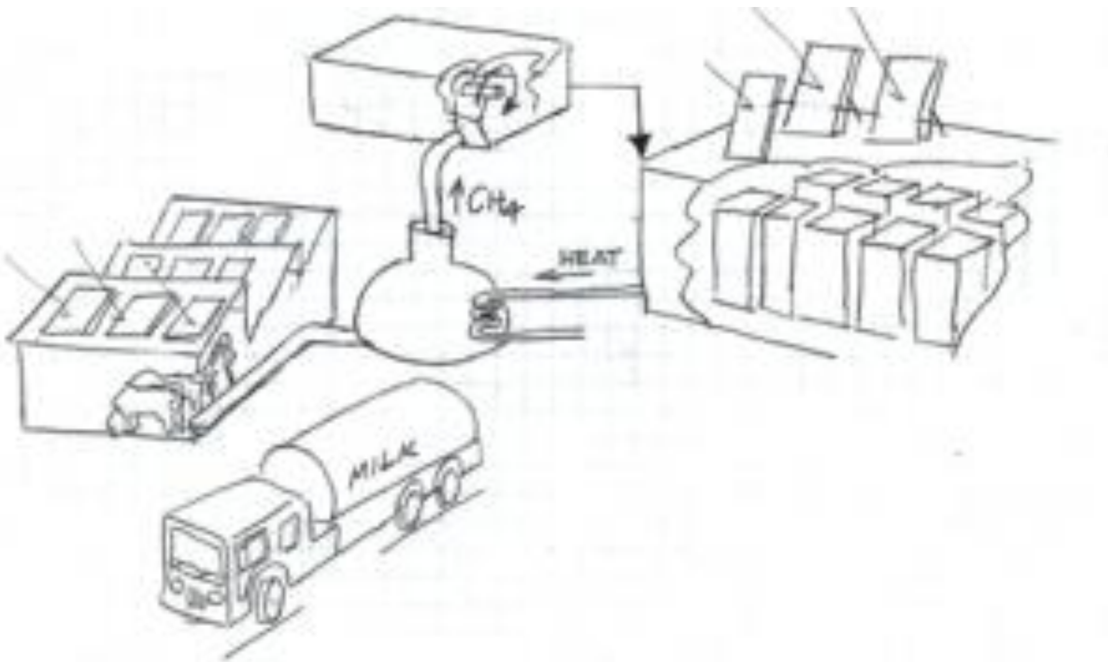
...seems impossible!

Lots of crazy & creative ideas!



A Google patent

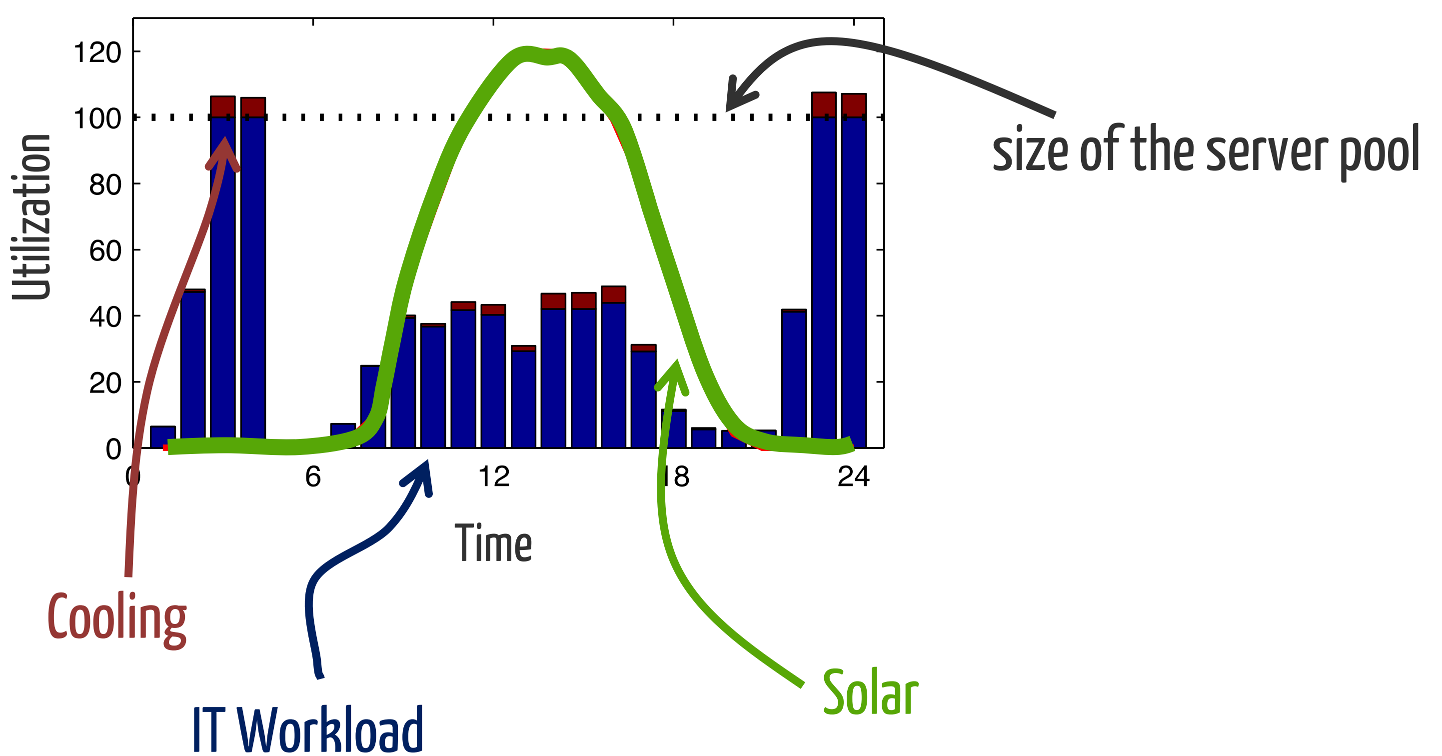
A Microsoft project



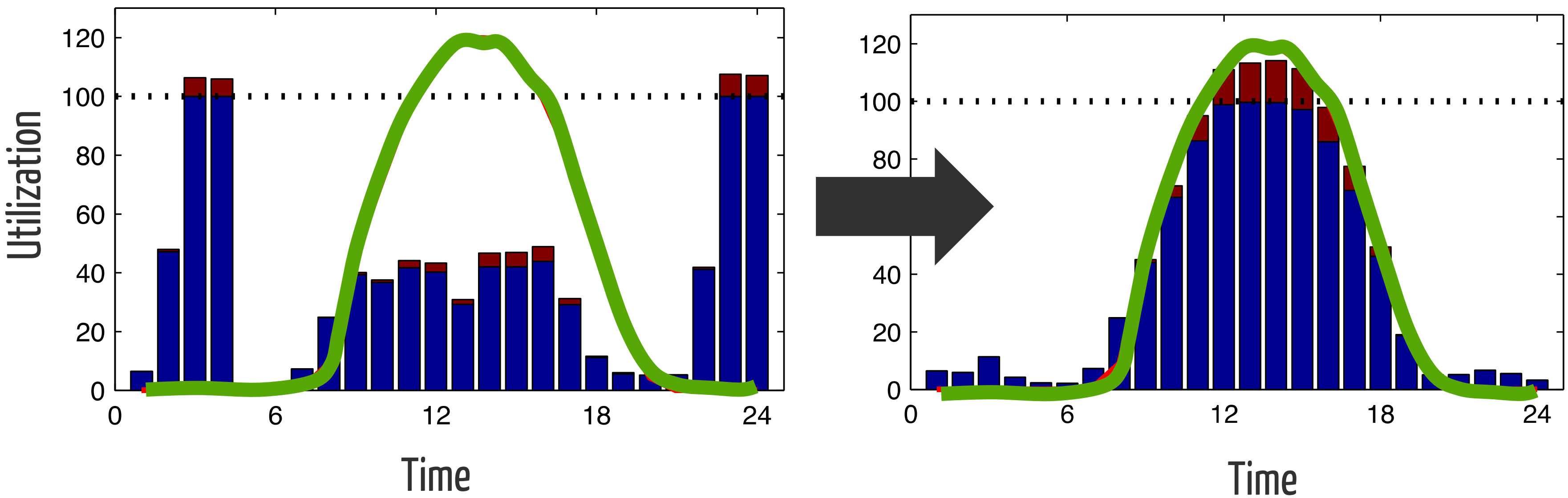
Drawing by Chandrakant Patel (Chief Engineer, HP)

...but all of these run into the same challenge

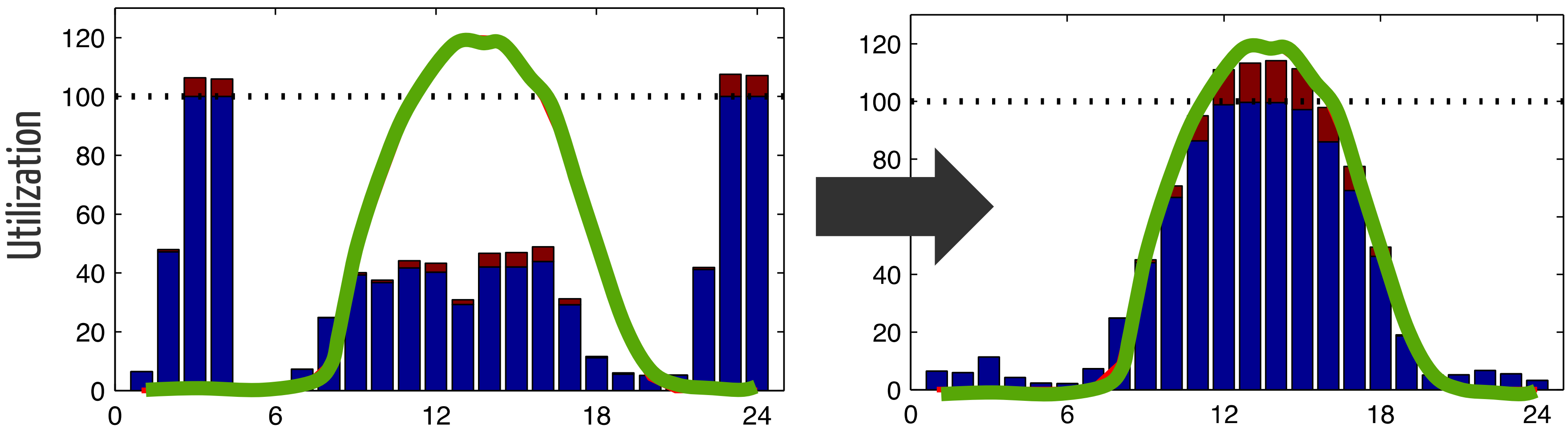
The challenge: Supply ≠ Demand



The challenge: Supply ≠ Demand



The challenge: Supply ≠ Demand



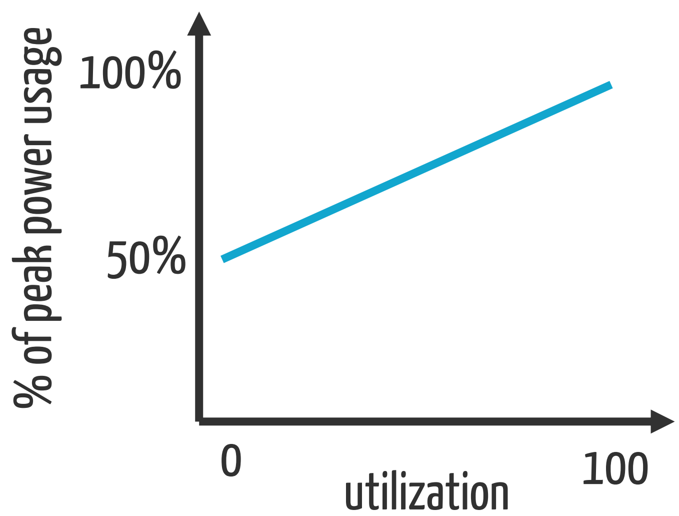
Energy Storage

&

Demand Response

How?

Servers use power even when idle



Servers use power even when idle & turning them off incurs significant cost

Turning a server off ≈ running a server for 30-60min

Servers use power even when idle & turning them off incurs significant cost
& you don't know future workloads
& predicting wind/solar is impossible
& data needs to be accessible
& not all workloads can be deferred
& ...

Core challenge: How can you know when to turn a server off (without knowing the future) ?

Does anyone ski?

Rent or Buy?

$20/day to rent or $200 to buy

Rent or Buy?

$20/day to rent or $200 to buy

Computer scientists are pessimists,
they always imagine the worst-case....

Can you make sure that you never spend too much more than optimal?

Buy right away?

Rent or Buy?

$20/day to rent or $200 to buy

Computer scientists are pessimists,
they always imagine the worst-case....

Can you make sure that you never spend too much more than optimal?

Buy right away?

Always rent?

Rent or Buy?

$20/day to rent or $200 to buy

Computer scientists are pessimists,
they always imagine the worst-case....

Can you make sure that you never spend too much more than optimal?

Buy right away?
Always rent?
Rent for a while and then buy... but for how long?

Keep on Turn off

~~Rent~~ or ~~Buy~~?

$20/day to rent or $200 to buy

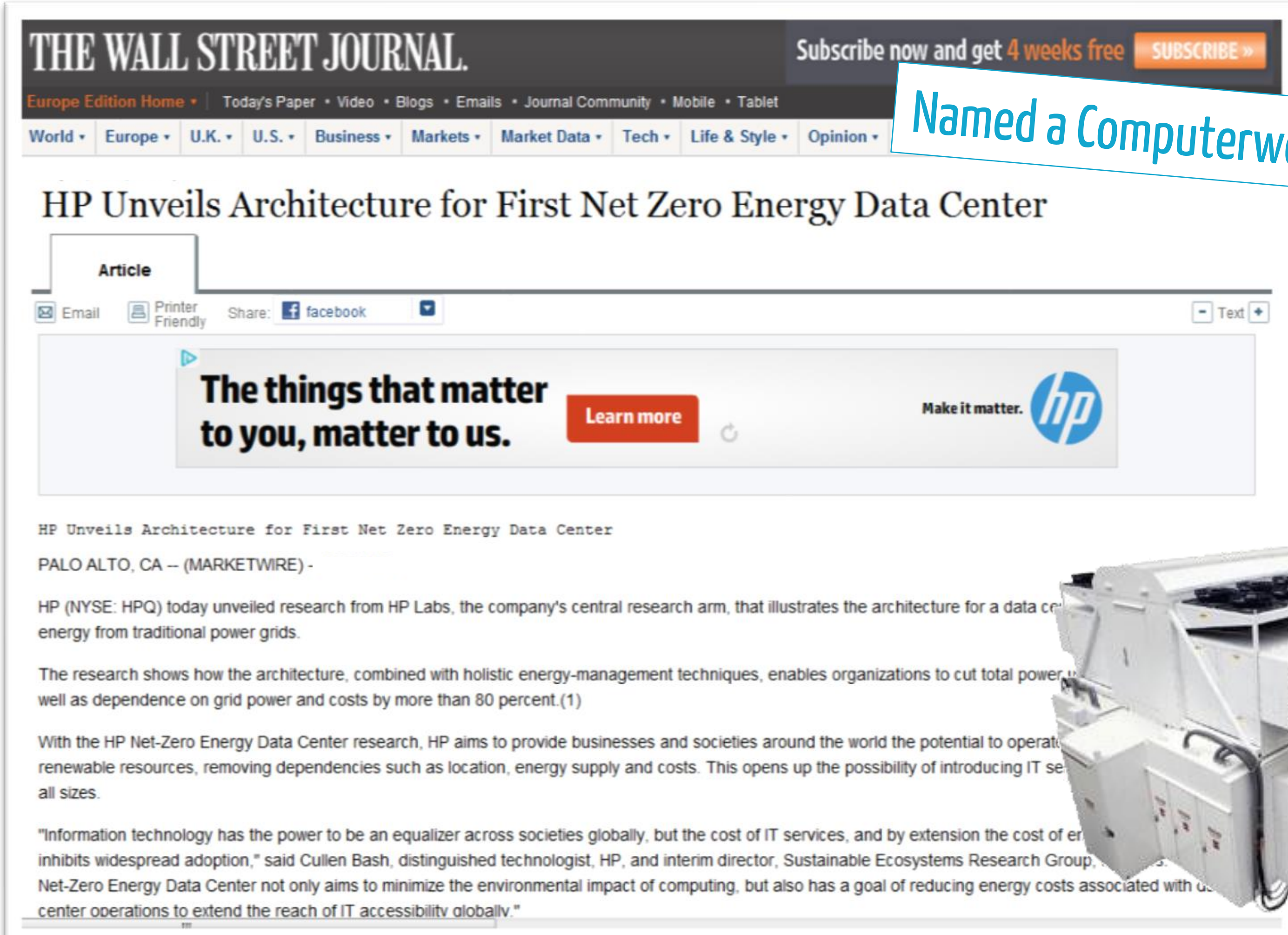
The optimal algorithm:
Rent until you've spent as much money as it costs to buy, then buy.

You're guaranteed to spend no more than twice the optimal
...which is the best possible for any "online" algorithm

...unless I give you a coin!

Years of research pass...

Net Zero Data Center Architecture & Eco Pod



THE WALL STREET JOURNAL.

HP Unveils Architecture for First Net Zero Energy Data Center

HP Unveils Architecture for First Net Zero Energy Data Center

PALO ALTO, CA -- (MARKETWIRE) -

HP (NYSE: HPQ) today unveiled research from HP Labs, the company's central research arm, that illustrates the architecture for a data c energy from traditional power grids.

The research shows how the architecture, combined with holistic energy-management techniques, enables organizations to cut total power well as dependence on grid power and costs by more than 80 percent.(1)

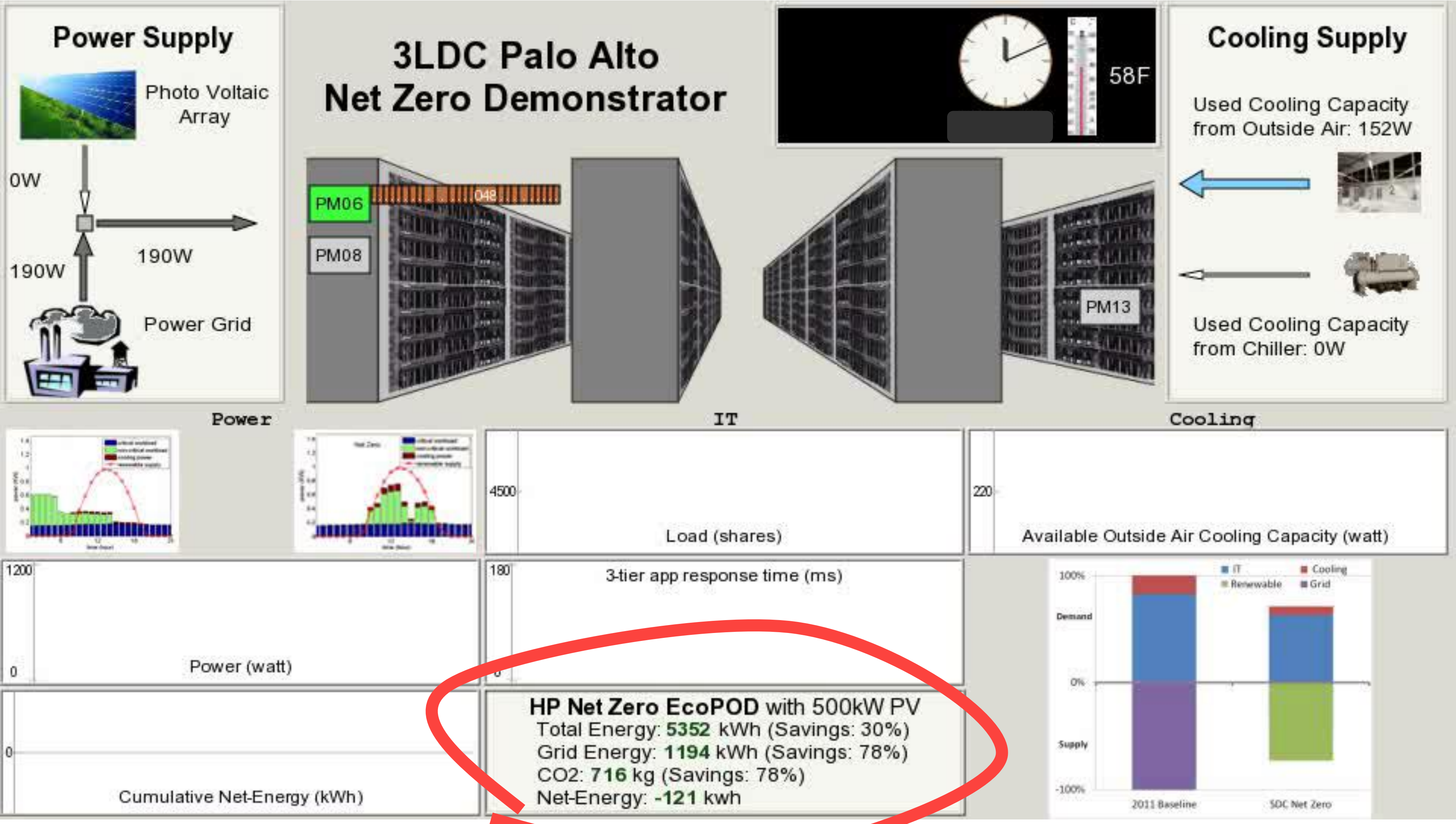
With the HP Net-Zero Energy Data Center research, HP aims to provide businesses and societies around the world the potential to operat renewable resources, removing dependencies such as location, energy supply and costs. This opens up the possibility of introducing IT se all sizes.

"Information technology has the power to be an equalizer across societies globally, but the cost of IT services, and by extension the cost of e inhibits widespread adoption," said Cullen Bash, distinguished technologist, HP, and interim director, Sustainable Ecosystems Research Group, Net-Zero Energy Data Center not only aims to minimize the environmental impact of computing, but also has a goal of reducing energy costs associated with d center operations to extend the reach of IT accessibility globally."

Named a Computerworld Honors Laureate

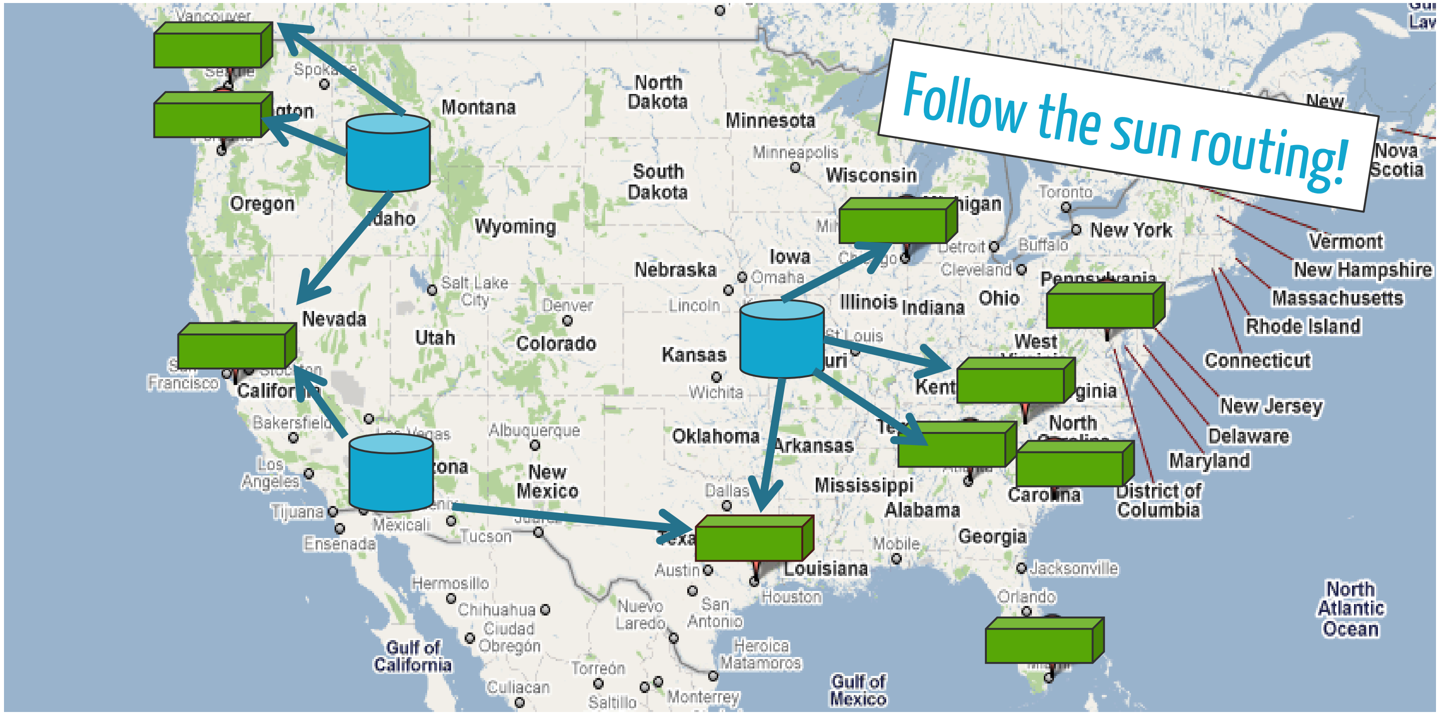
HP collaborators: Yuan Chen, Cullen Bash, Martin Arlitt, Daniel Gmach, Zhikui Wang, Manish Marwah and Chris Hyser

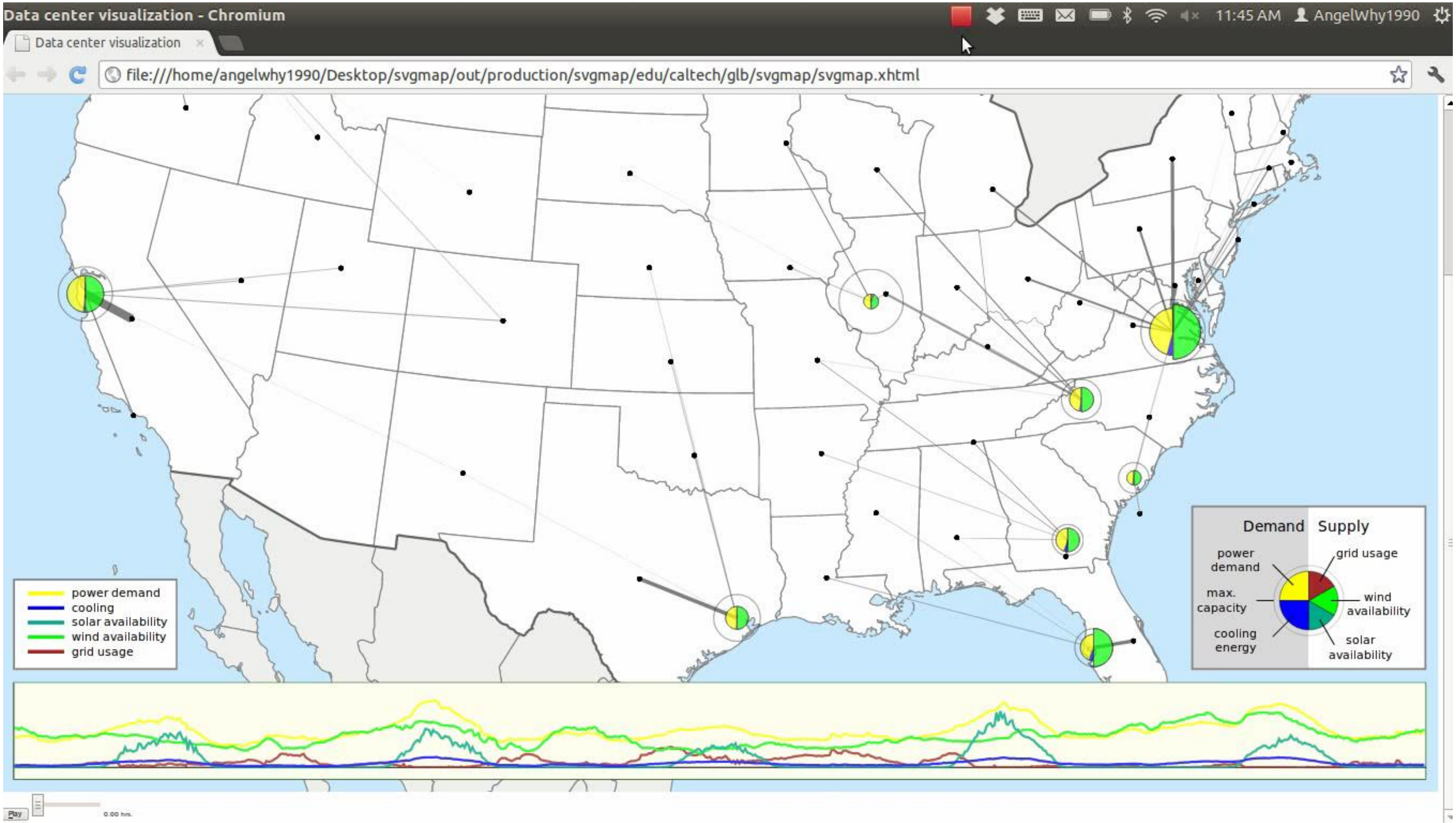
Net Zero Data Center Architecture



HP collaborators: Yuan Chen, Cullen Bash, Martin Arlitt, Daniel Gmach, Zhikui Wang, Manish Marwah and Chris Hyser

...and this extends beyond a single data center





http://www.youtube.com/watch?v=IhyqYVsTM_8&feature=youtu.be

KEY ISSUES FOR DATA CENTERS

1. $\boxed{\text{Energy usage}}$
2. Ease of programming
3. Reliability
4. Resource allocation (Performance issues)

KEY ISSUES FOR DATA CENTERS

1. Energy usage
2. $\boxed{\text{Ease of programming}}$
3. Reliability
4. Resource allocation (Performance issues)

The challenge

Traditional programming is serial, and
parallel programming is notoriously tricky.

…but we need to make use of 1000s of parallel machines!
…and we need it easy enough to roll out software quickly!

Google's answer: MapReduce

Created in 2004 (Jeffery Dean and Sanjay Ghemawat)

Key observation:

Google was solving "embarrassingly parallel" problems.

→ No dependency among data:

Each piece of the data can be processed independently and then combined

They didn't need to solve the general "parallel programming problem"!

Google's answer: MapReduce

Created in 2004 (Jeffery Dean and Sanjay Ghemawat)

Solution was inspired by LISP

→ **map**(function, set of values)
Applies the function to each value in the set
e.g. map(length, ((x),(x,y),(z),(x,y,z))) → (1,2,1,3)

→ **reduce**(function, set of values)
Combines the values using the function
e.g. reduce(+, (1,2,1,3)) → 7

TotalLength(list of lists) = reduce(+,map(length, list of lists))

Google's answer: MapReduce

Created in 2004 (Jeffery Dean and Sanjay Ghemawat)

system provides

programmer provides

→ **map**(function, set of values)
Applies the function to each value in the set
e.g. map(length, ((x),(x,y),(z),(x,y,z))) → (1,2,1,3)

→ **reduce**(function, set of values)
Combines the values using the function
e.g. reduce(+, (1,2,1,3)) → 7

TotalLength(list of lists) = reduce(+,map(length, list of lists))

Google's answer: MapReduce

Created in 2004 (Jeffery Dean and Sanjay Ghemawat)

The reality is almost that simple…

Programmers don't need to worry about

→ Data distribution

→ Load balancing across servers

→ Fault tolerance

…

But, MapReduce is not the same as LISPs "map" and "reduce"…

MapReduce: List of (key,value) pairs → List of (key,value) pairs

MapReduce(list of (key,value) pairs)

= reduce(Reduce,Collect(map(Map,List of (key,value) pairs)))

Defined by the programmer

MapReduce: List of (key,value) pairs → List of (key,value) pairs

MapReduce(list of (key,value) pairs)
= reduce(Reduce,Collect(map(Map,List of (key,value) pairs)))

Defined by the programmer

Map Phase

List of (key,value) → List of (key,value)
"map"s a bunch of parallel Map() calls over the list
Map:(key,value)→(key,value)

MapReduce: List of (key,value) pairs → List of (key,value) pairs

MapReduce(list of (key,value) pairs)
= reduce(Reduce,Collect(map(Map,List of (key,value) pairs)))

Defined by the programmer

Collector Phase
a.k.a. Copy/Shuffle

List of (key,value) → List of (key,list of values)
Collects all outputs from Map(), sorts by key, and makes a list of values for each unique key

MapReduce: List of (key,value) pairs → List of (key,value) pairs

MapReduce(list of (key,value) pairs)
  = reduce(Reduce,Collect(map(Map,List of (key,value) pairs)))

Defined by the programmer

Reduce Phase   List of (key,list of values) → List of (key,value)
"reduce"s each list using parallel Reduce() calls
Reduce:(key,list of values)→(key,value)

MapReduce: List of (key,value) pairs → List of (key,value) pairs

Input	List of (key,value) pairs
Map Phase	List of (key,value) → List of (key,value) "map"s a bunch of parallel Map() calls over the list Map:(key,value)→(key,value)
Collector Phase a.k.a. Copy/Shuffle	List of (key,value) → List of (key,list of values) Collects all outputs from Map(), sorts by key, and makes a list of values for each unique key
Reduce Phase	List of (key,list of values) → List of (key,value) "reduce"s each list using parallel Reduce() calls Reduce:(key,list of values)→(key,value)
Output	Outputs a list of all the (key,value) pairs output by the Reduce() calls

MapReduce: List of (key,value) pairs → List of (key,value) pairs

MapReduce(list of (key,value) pairs)

= reduce(Reduce,Collect(map(Map,List of (key,value) pairs)))

Defined by the programmer

MapReduce: List of (key,value) pairs → List of (key,value) pairs

MapReduce(list of (key,value) pairs)
  = reduce(Reduce,Collect(map(Map,List of (key,value) pairs)))

Evaluated in parallel

MapReduce: List of (key,value) pairs → List of (key,value) pairs

MapReduce(list of (key,value) pairs)

= reduce(Reduce,Collect(map(Map,List of (key,value) pairs)))

Runs as each Map() finishes

MapReduce: List of (key,value) pairs → List of (key,value) pairs

MapReduce(list of (key,value) pairs)

= reduce(Reduce,Collect(map(Map,List of (key,value) pairs)))

Evaluated in parallel

MapReduce Examples…

Average Income
Input: List of (SSN,(City,Income,…)) for individuals
Output: List of (City, Average income) for all cities

```
Map(SSN,(City,Income,…)) {
     return (City,Income)
}
```

After collector you have
List of (City, (Income1,Income2,…))

```
Reduce(City,list of Incomes) {
    return (City, Average(list of incomes))
}
```

→ Easy to get basic database functionality

MapReduce Examples…

Word count

Input: Large number of text documents

Output: List of (word, #occurrences) for all words in the documents

```
Map(filename,filecontents) {
   words[] = split  filecontents by " ".
   for each word in words {
       return (word,1)
   }
}
```

After collector you have

List of (word, (1,1,…,1))

```
Reduce(word,list of ones) {
   return (word, length(list of values))
}
```

MapReduce Examples…

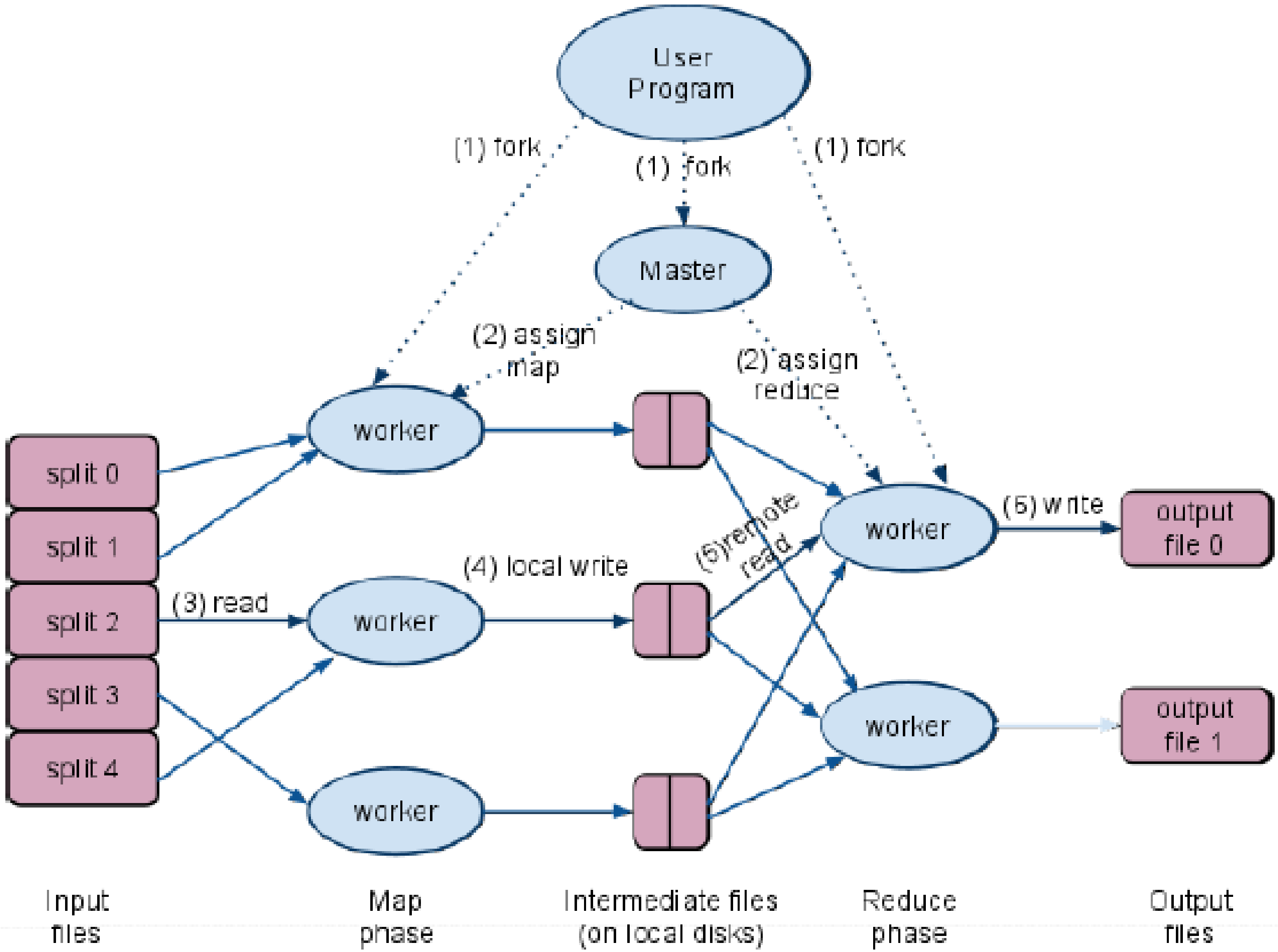
You'll do lots more examples on your HW!

MapReduce: List of (key,value) pairs → List of (key,value) pairs

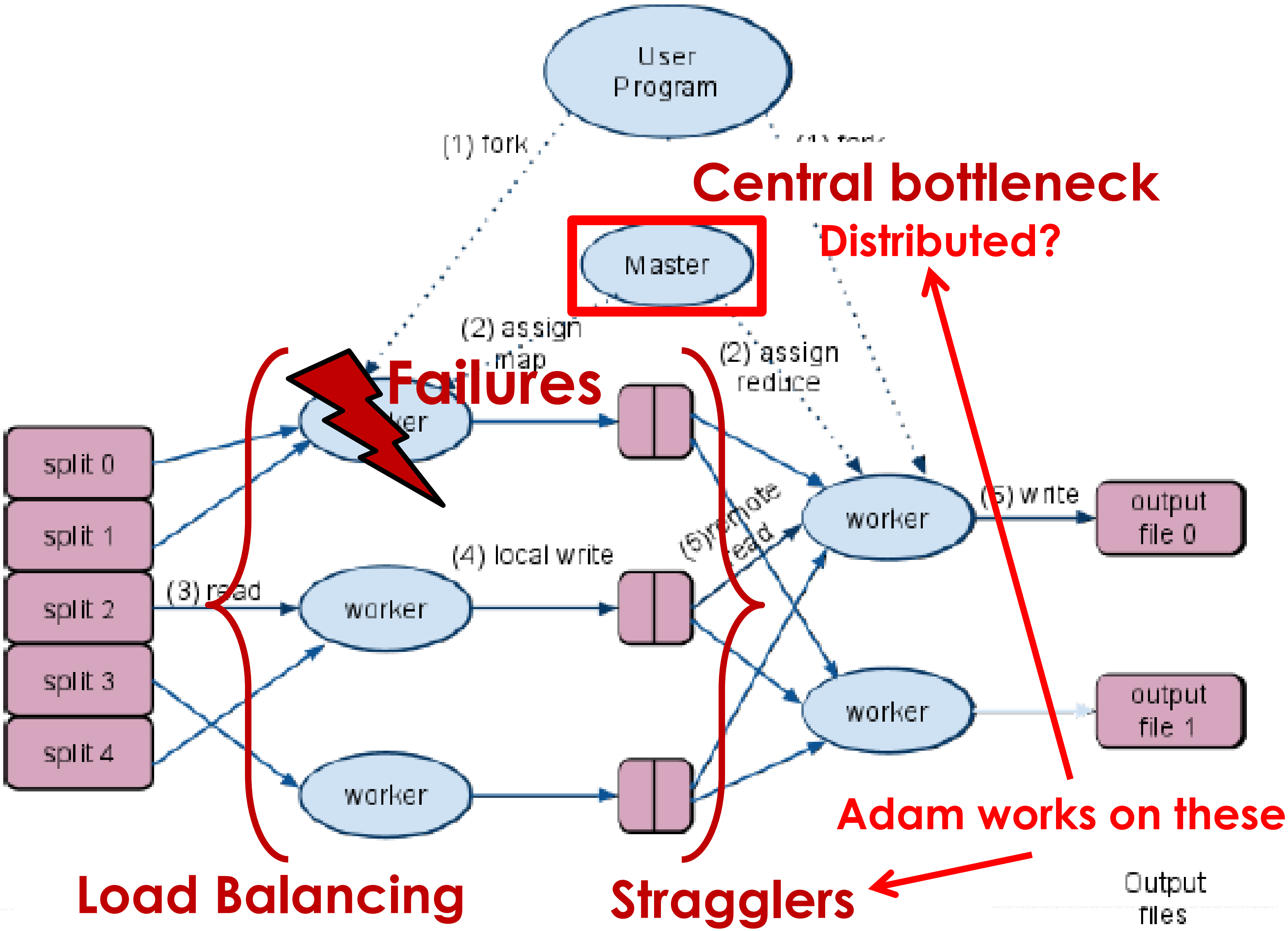
MapReduce(list of (key,value) pairs)

= reduce(Reduce,Collect(map(Map,List of (key,value) pairs)))

MapReduce Execution Overview



Practical issues



Google's answer: MapReduce

Created in 2004 (Jeffery Dean and Sanjay Ghemawat)

MapReduce inside Google

Google

Googlers' hammer for 80% of our data crunching

- Large-scale web search indexing
- Clustering problems for Google News
- Produce reports for popular queries, e.g. Google Trend
- Processing of satellite imagery data
- Language model processing for statistical machine translation
- Large-scale machine learning problems
- Just a plain tool to reliably spawn large number of tasks
  - e.g. parallel data backup and restore

The other 20%? e.g. Pregel

Google's answer: MapReduce

Created in 2004 (Jeffery Dean and Sanjay Ghemawat)

Use of MapReduce inside Google

Google

Stats for Month	Aug.'04	Mar.'06	Sep.'07
Number of jobs	29,000	171,000	2,217,000
Avg. completion time (secs)	634	874	395
Machine years used	217	2,002	11,081
Map input data (TB)	3,288	52,254	403,152
Map output data (TB)	758	6,743	34,774
reduce output data (TB)	193	2,970	14,018
Avg. machines per job	157	268	394
Unique implementations			
Mapper	395	1958	4083
Reducer	269	1208	2418

Google's answer: MapReduce

Created in 2004 (Jeffery Dean and Sanjay Ghemawat)

For us outside of google:

→ Apache Hadoop
(built to specs defined by Google)

→ Amazon Elastic MapReduce
(started from Hadoop implementation)

You'll use this for Rankmaniac!

Google's answer: MapReduce

Created in 2004 (Jeffery Dean and Sanjay Ghemawat)

Some controversy about MapReduce…

Often touted as "revolutionary"
…many distributed database folks disagree.

Key point – it's not a full solution for distributed software.

Google's answer: MapReduce

Created in 2004 (Jeffery Dean and Sanjay Ghemawat)

Google

	MPI	MapReduce	DBMS/SQL
What they are	A general parrellel programming paradigm	A programming paradigm and its associated execution system	A system to store, manipulate and serve data.
Programming Model	Messages passing between nodes	Restricted to Map/Reduce operations	Declarative on data query/retrieving; Stored procedures
Data organization	No assumption	"files" can be sharded	Organized datastructures
Data to be manipulated	Any	k,v pairs: string/protomsg	Tables with rich types
Execution model	Nodes are independent	Map/Shuffle/Reduce Checkpointing/Backup Physical data locality	Transaction Query/operation optimization Materialized view
Usability	Steep learning curve*; difficult to debug	Simple concept Could be hard to optimize	Declarative interface; Could be hard to debug in runtime
Key selling point	Flexible to accommodate various applications	Plow through large amount of data with commodity hardware	Interactive querying the data; Maintain a consistent view across clients

Google's answer: MapReduce

Created in 2004 (Jeffery Dean and Sanjay Ghemawat)

Some controversy about MapReduce…

Often touted as "revolutionary"
…many distributed database folks disagree.

Key point – it's not a full solution for distributed software.

In fact, plain-vanilla map-reduce is already "old"
…many current cluster frameworks extend to
more complex DAGs of tasks now.

KEY ISSUES FOR DATA CENTERS

1. Energy usage
2. Ease of programming
3. Reliability
4. Resource allocation (Performance issues)

Also very important

That’s it!