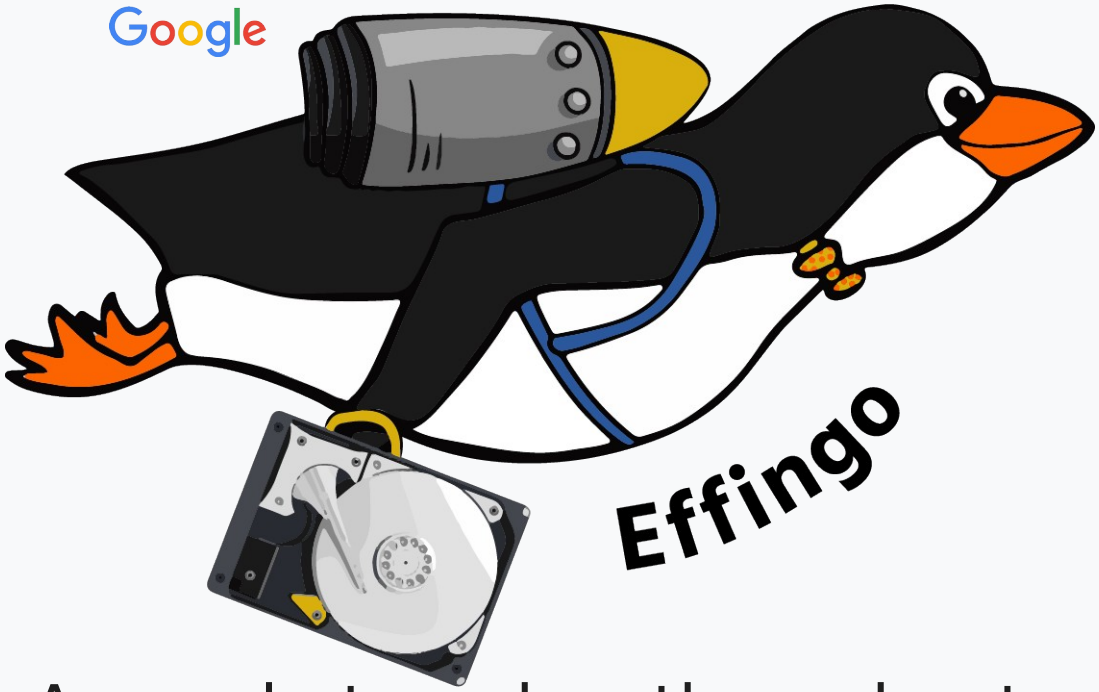


An exabyte a day: throughput-oriented, large scale, managed data transfers with Effingo

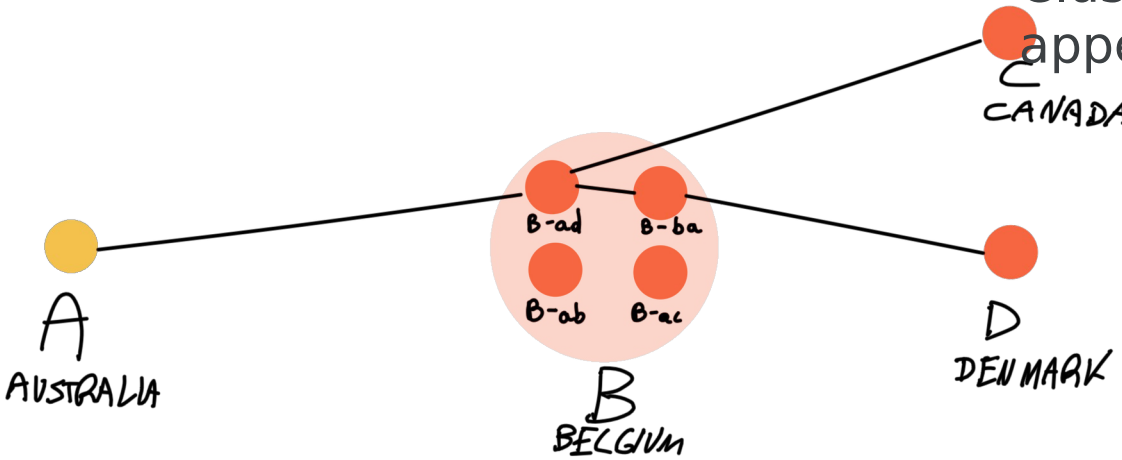
Michał Zasadziński, PhD
Engineering Manager, Google

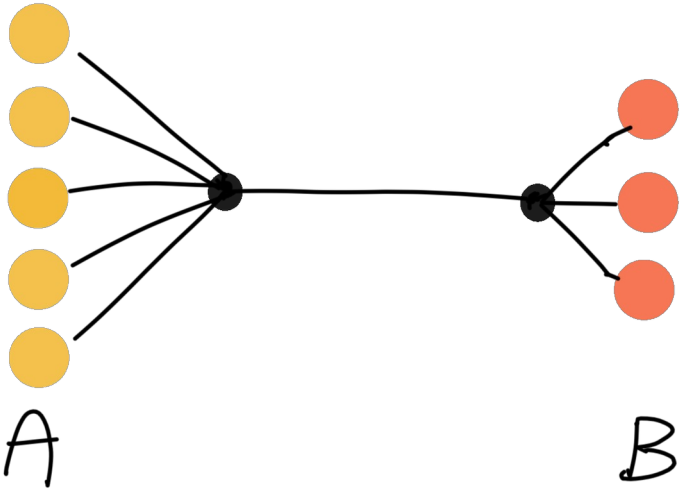
How to copy files ?

- Dynamic system of 100+ data centers across many countries and continents
  - various network link prices
  - structure and the demand is dynamic
  - heterogeneous resources: CPUs, storage, network
  - Cluster-level failure domains - Transmission cannot depend on any third "router" cluster.
- We need to give throughput guarantees
  - users and their copies with different priorities
  - SLO

Google Cloud Platform has 35+ geo regions, thus 100+ clusters.

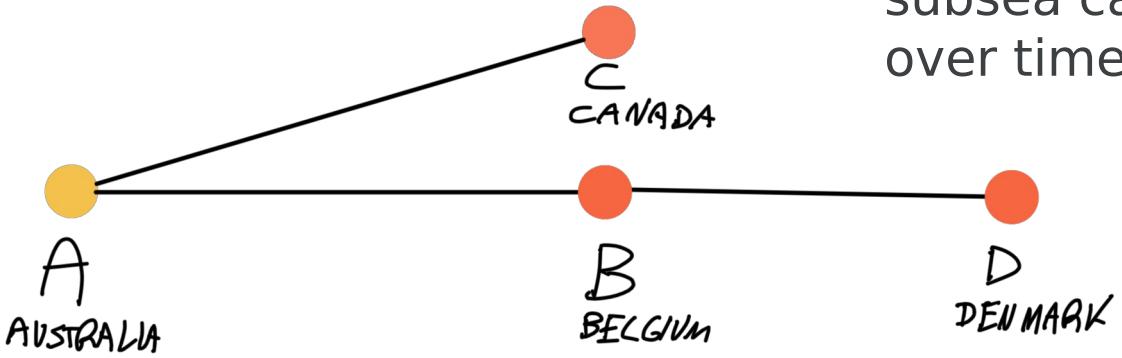
Dynamic infrastructure: Clusters grow, shrink, appear and disappear.





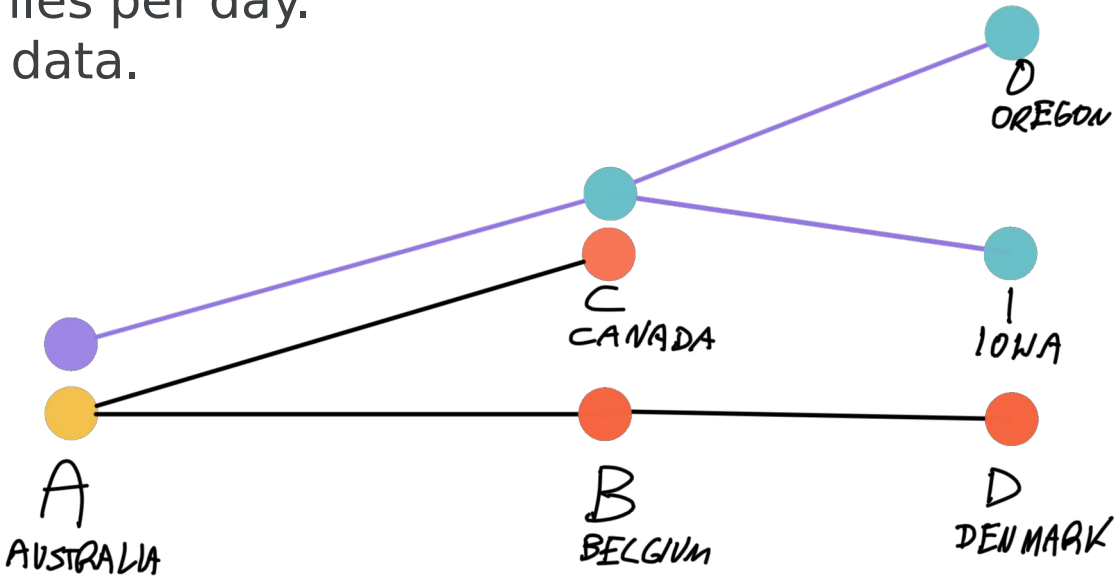
Each file is stored on **multiple physical servers**.

File **replication/erasure coding** might **differ** between locations and filesystems.



Routing to reduce transfers on expensive subsea cables. May change over time.

Thousands of users.
Billions of files per day.
Exabyte of data.

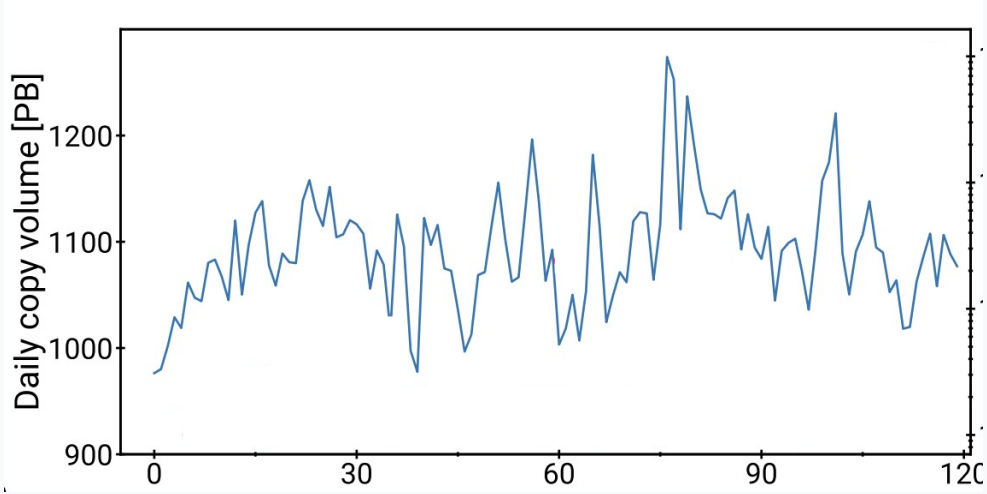


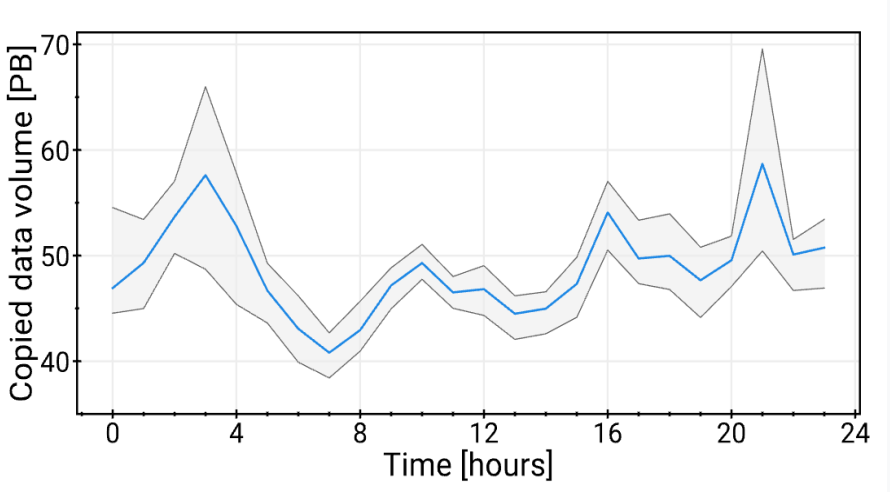
Many transfers in parallel: **performance interference** on egress, ingress, IO and network links.

Effingo

- Operates on all Google clusters. Clients can be anywhere, same with data.
- Wide spectrum of users
  - batch
  - replication
  - latency sensitive
  - many others, e.g., with own scheduling
- Transfer files between clusters with cluster level failure domain
- Copy as fast as possible without exhausting resources
- Control plane (copy request orchestration) and data plane (bytes transfer)
- Supports a variety of file system types

1 EiB per day and growing

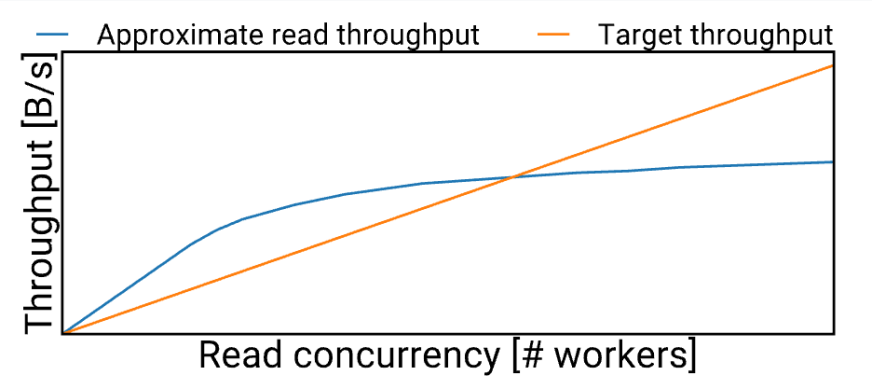




- 10k+ users, dozens of clusters, typical day with peaks up to 21 TiB/s
- high scalability in every cluster and every Effingo's module
- peaks up to 400k aggregated; 200k completed files per second per cluster

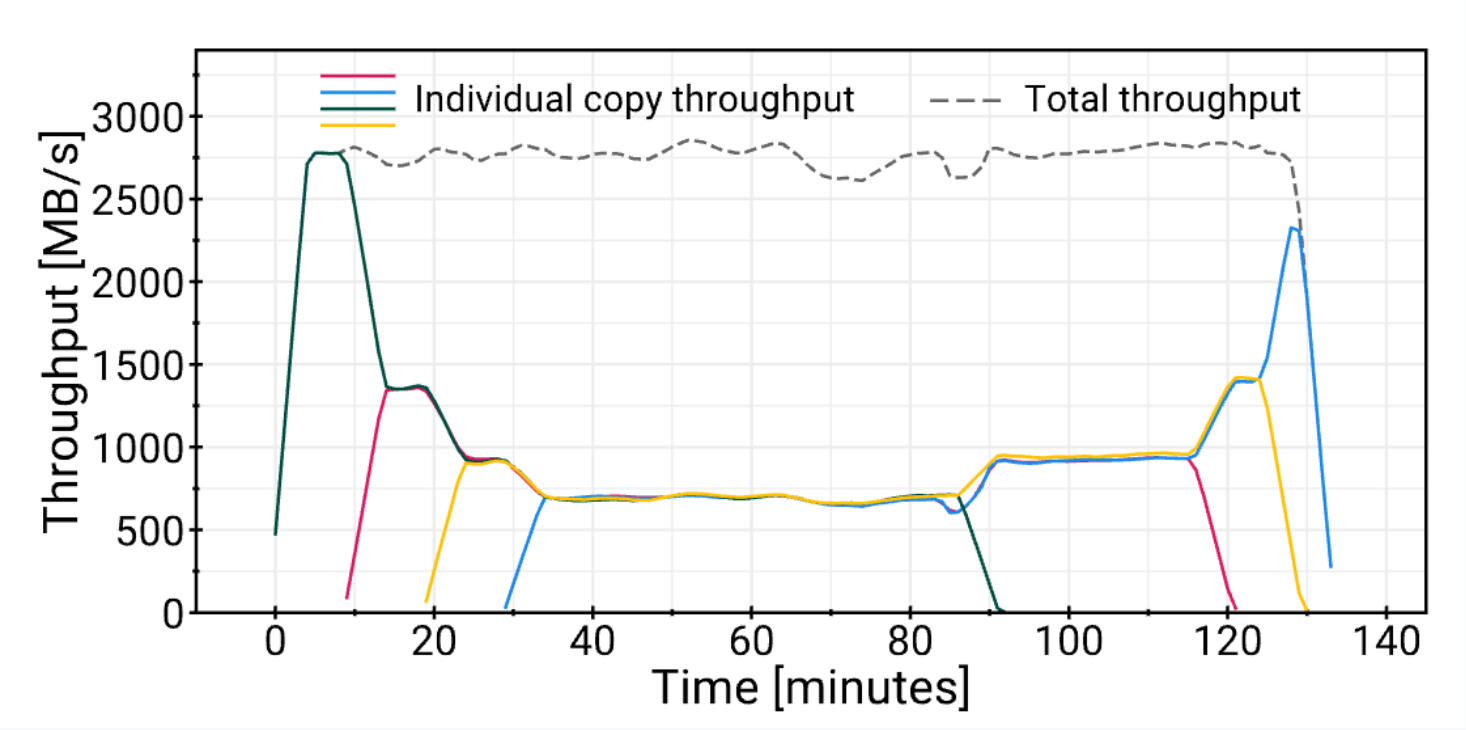
Maximum throughput under SLO

- Find optimal parallelism with two algorithms
- Preventing resource stranding: converge to the most congested resources used in the transfer; react to errors
- Regulating throughput: P controller



Regulating throughput by adjusting the number of workers. When the number of workers increases, the resulting bandwidth increases too, but only to a certain threshold. The control loop algorithm increases concurrency when throughput is greater than target and reduces otherwise

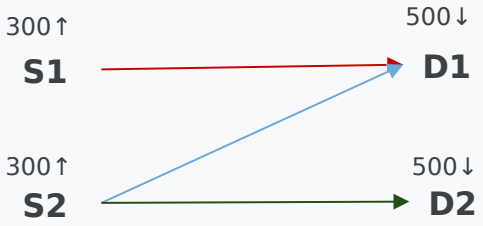
Fair resource distribution: competing transfers

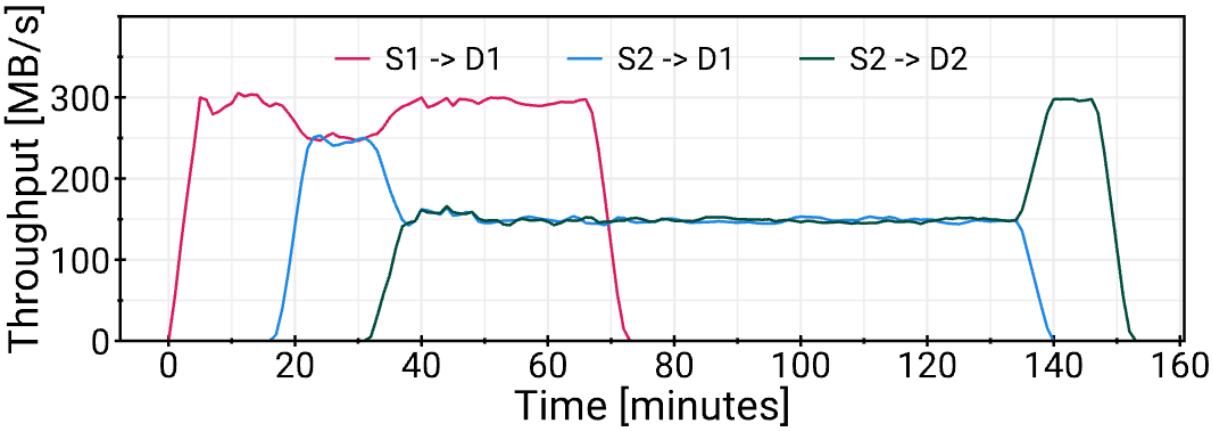


Resource sharing with four equal priority operations starting at times 0, +10m, +20m, and +30m.
Each quickly gets an equal share.
Sharing is efficient as the total throughput is maximized regardless of the number of

Fair and efficient resource distribution

Network topology:





Effingo achieves fairness and efficiency without global coordination.
Clusters S1, S2 output is 300MB/s, while D1, D2 input is 500MB/s.
When S2->D1 starts, fair sharing of D1 input limits S1->D1 to 250MB/s.

Then, when S2->D2 starts, S2->D1 becomes constrained by sharing S2 output, resulting in roughly 150MB/s, which enables S1->D1 to recover to 300MB/s.

An exabyte a day: throughput oriented, large scale, managed data transfers with Effingo

SIGCOMM Paper authors: Ladislav Pápay, Jan Pustelnik, **Krzysztof Rzadca**, Beata Strack, Paweł Stradomski, Bartłomiej Wołowiec, **Michał Zasadziński**

Cluster-level **decentralization** for cluster-level failure domains

Massive **scale**: billions of independently transmitted files with different priorities

Heterogeneity hardware, network, software stacks

Dynamic changes: Clusters come and go, throughput varies.