Power- and Fragmentation-aware Online Scheduling for GPU Datacenters

F.Lettich E.Carlini F.M.Nardini R.Perego S.Trani

Institute of Information Science and Technologies "Alessandro Faedo"
National Research Council of Italy

IEEE CCGrid, 19-22 May 2025, Tromsø, Norway

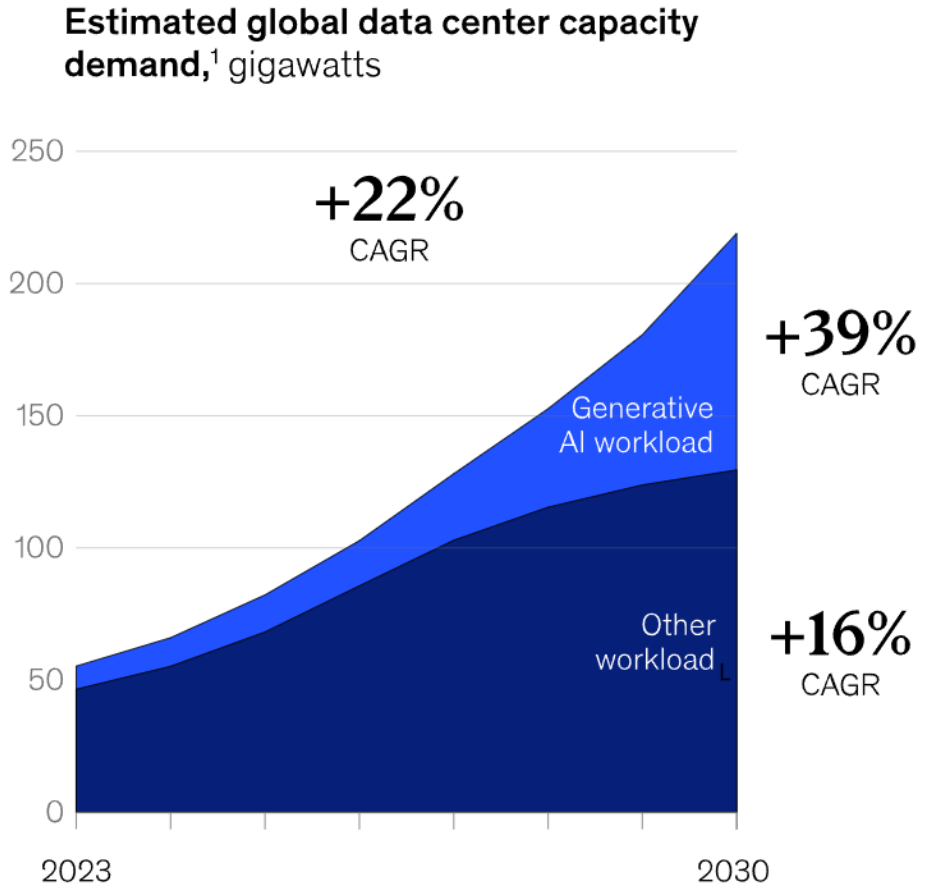
HPC lab

EFRA EXTREME FOOD RISK ANALYTICS

FAIR Future Artificial Intelligence Research

Introduction / 1

- The rapid growth of AI, ML, and in recent times, **large language models** and **test-time compute**, is leading to a **massive increase** in **size** and **number** of GPU datacenters.



https://www.mckinsey.com/industries/technology-media-and-telecommunications/our-insights/investing-in-the-rising-data-center-economy
https://www.mckinsey.com/industries/technology-media-and-telecommunications/our-insights/ai-power-expanding-data-center-capacity-to-meet-growing-demand

- **Serious challenges**: soaring **power consumption**, **operational costs, environmental impacts**…

Introduction / 2

- We consider the **scenario** from **[1]**, based on a **real-world Alibaba production-grade** GPU datacenter. Very **interesting features**…

- The **datacenter** processes **hybrid workloads:** mix of **training** and **inference** tasks.

- **Online scheduling**: tasks are assigned to a node **as they arrive**. **No knowledge** of **future arrivals**.

- **Support** for **GPU-sharing** tasks: tasks that use a **fraction** of **a GPU**'s resources, and **are allowed** to (time-)**share it** with other GPU-sharing **tasks**.

[1] Weng, Qizhen, et al. "*Beware of fragmentation: Scheduling GPU-Sharing workloads with fragmentation gradient descent*". USENIX ATC 2023.

Introduction / 3

- **[1]** focused on **minimizing** the **fragmentation** of **GPU resources ($F_{datacenter}$): due** to **GPU-sharing**, it becomes **critical** when a **datacenter** is **near saturation**.

- In our work, **alongside** GPU fragmentation, we **consider** an **additional dimension**: **power consumption ($P_{datacenter}$). Reducing** consumption is **possible** when **not close** to saturation.

- Get this **optimization problem**:

$$\arg\min_{t \to n} \; \alpha \cdot P_{datacenter} + (1 - \alpha) \cdot F_{datacenter}$$

[1] Weng, Qizhen, et al. "*Beware of fragmentation: Scheduling GPU-Sharing workloads with fragmentation gradient descent*". USENIX ATC 2023.

Outline

- Introduction.
- **Contributions**
- Experimental evaluation
- Final considerations

Contributions: preliminaries

- Modeling **node resources**: **unallocated** (R_n) and **allocated** (Ra_n) resource vectors.

$$R_n = \langle R_n^{CPU}, R_n^{MEM}, R_{n,1}^{GPU}, \ldots, R_{n,G_n}^{GPU} \rangle$$

NOTE: R^{CPU} is the number of virtual CPUs. Assume 2 virtual CPUs per CPU physical core.

- Modeling **tasks** (**demands** and **constraints**):

$$D_t = \langle D_t^{CPU}, D_t^{MEM}, D_t^{GPU} \rangle, \; C_t = \{C_t^{CPU}, C_t^{GPU}\}$$

NOTE (simplification): a task either demands a **fraction** of the resources of a **single GPU**, or **entire 1…n** GPUs, but **not both**.

Contributions: power consumption estimation model

- Simple **model** to **estimate** the **power consumption** of **CPUs** and **GPUs in a node**; takes into consideration **CPU** and **GPU models**.

$$p_{CPU}(n) = p_{max}(type_{CPU}(n)) \cdot \left\lceil \frac{Ra_n^{CPU}}{2 \cdot ncores(type_{CPU}(n))} \right\rceil + p_{idle}(type_{CPU}(n)) \cdot \left\lfloor \frac{R_n^{CPU}}{2 \cdot ncores(type_{CPU}(n))} \right\rfloor$$

$$p_{GPU}(n) = \sum_{g=1}^{G_n} \begin{cases} p_{max}(type_{GPU}(n)) \text{ if } Ra_{n,g}^{GPU} > 0 \\ p_{idle}(type_{GPU}(n)) \text{ otherwise} \end{cases}$$

$$P_{datacenter} = \sum_{n=1}^{N} p_{CPU}(n) + p_{GPU}(n)$$

Contributions: our PWR scheduling policy

- We consider the customized version Alibaba's ***open-simulator*** from [1]. It is a GPU datacenter simulator.

- We **customized** a little bit further the simulator to **support power telemetry**.

- Developed a new Kubernetes **scoring plugin**, **PWR**. **Uses** the power estimation **model** to score nodes.

- When a task must be scheduled: **PWR scores** nodes **according to** the **Δs** in **power consumption** we'd have by **hypotetically scheduling** the task on them.

Contributions: combining PWR with FGD

- **Hold on: what about GPU fragmentation**?!

- **Leverage** the **Kubernetes scheduling framework: linearly combine PWR**'s and **FGD**'s scores. FGD is the scoring **plugin** from **[1]** targeting **GPU fragmentation**.

- **Crucial parameter: weight** given to PWR's and FGD's scores, i.e., **α (1-α)**.

- We get an **overall** scheduling policy that, with **appropriate αs**, offers **balanced tradeoffs** between the two goals.

Outline

- Introduction
- Contributions
- **Experimental evaluation**
- Final considerations

Experimental Setup / 1

- Conducted many simulations with **open-simulator**.

- Simulations **based** on the **Alibaba's production-grade GPU datacenter** from their 2023 GPU trace dataset [1]:
  - 1213 **nodes** (310 without any GPU);
  - 107,018 **virtual CPUs**;
  - 6,212 **GPUs** of different models (see table below);
  - **CPU** models N/A; we assume **Intel Xeon ES-2682 V4**.

TABLE II: GPU models in the trace data.

GPU model	Amount	Power idle (W)	TDP (W)
V100M16	195	30	300
V100M32	204	30	300
P100	265	25	250
T4	842	10	70
A10	2	30	150
G2 (A10)	4,392	30	150
G3 (A100)	312	50	400

Experimental Setup / 2

- Generate **workloads** using the **2023 Alibaba GPU trace dataset** [1].

- The dataset contains **4 different types** of traces (more on them on the next slides).

- Given a trace: **compute** the **probability distribution** of the **task profiles**. Then, **sample** from the distribution.

- **Fill** the datacenter **until arrived tasks** have requested **1.2 times** the datacenter's GPU available resources.

Experimental Setup / 3

- We **evaluate PWR** and **PWR+FGD** against **plain FGD [1]** and **several other competitors** that **support GPU-sharing tasks** and are readily available in open-simulator.

- Comparisons are done according to **two metrics**:
  - **Power savings vs FGD**.
  - **GPU Resource Allocation Ratio**:
    GPU resources **allocated** to **scheduled tasks** / GPU resources **requested** by **arrived tasks.**

- **Second** metric is a **proxy** for **GPU fragmentation**: the closer to 1, the better.

Experimental evaluation: Power consumption baseline

- We start by generating workloads from the **default trace**; we show FGD's power consumption as **baseline reference**.

TABLE I: Distribution of tasks in the Default trace.

GPU Request per Task	0	(0, 1)	1	2	4	8
Task Population (%)	13.3	37.8	48.0	0.2	0.2	0.5
Total GPU Reqs. (%)	0	28.5	64.2	0.5	1.0	5.8

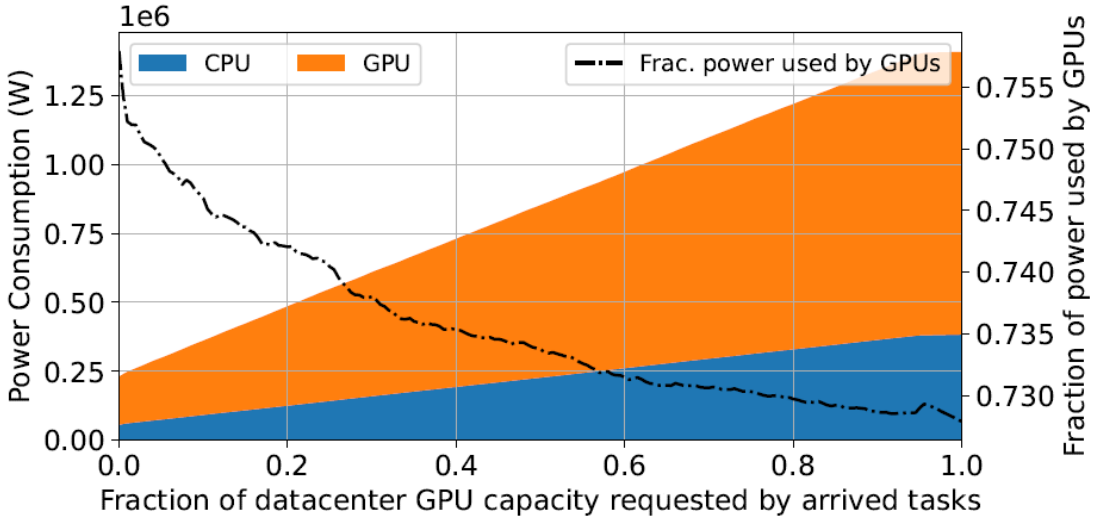
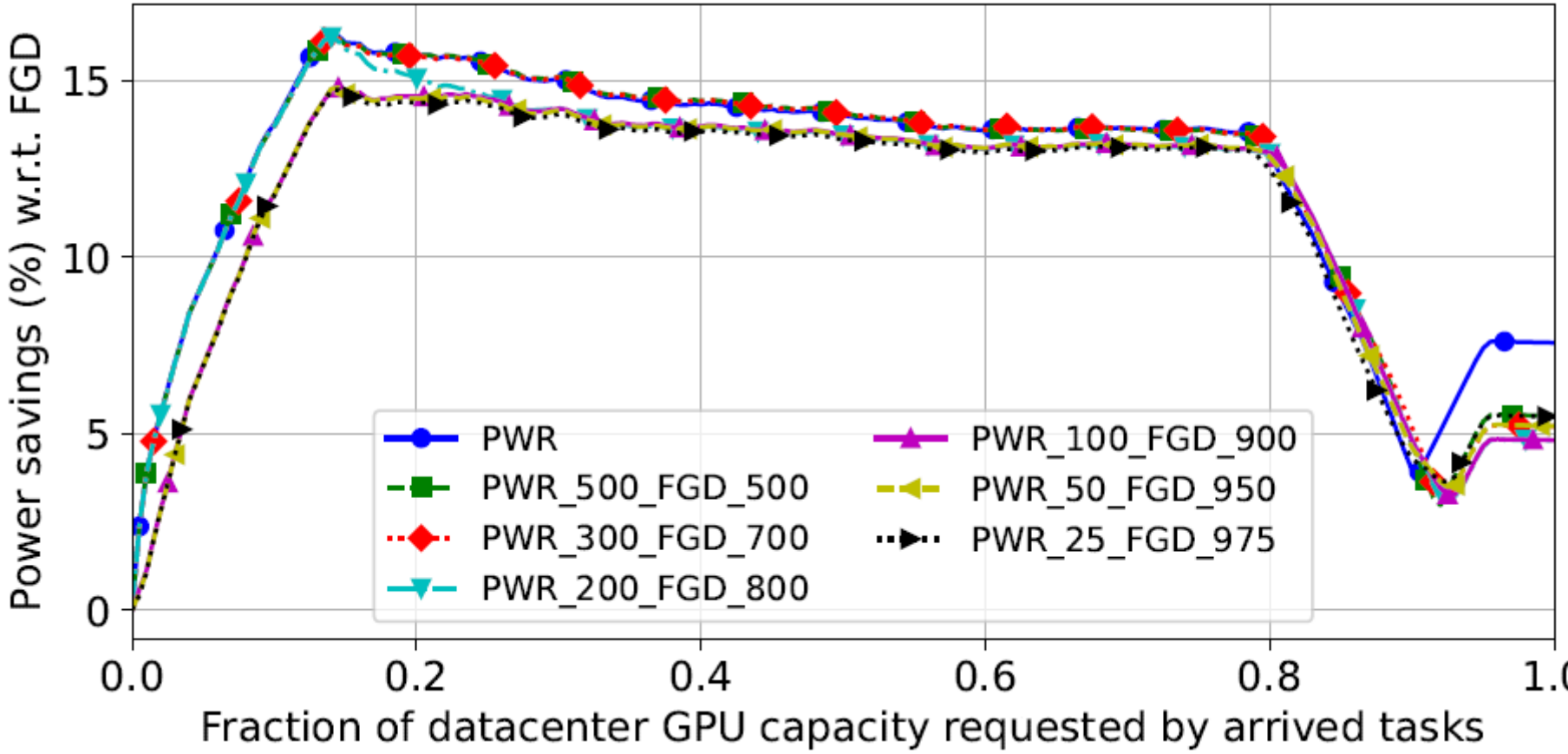


Fig. 1: FGD EOPC (in MW) for workloads from the Default trace, with stacked CPU and GPU components. The dashed line shows the fraction of GPU power (see right-hand y-axis).

Experimental evaluation: Comparing PWR and PWR+FGD vs FGD / 1

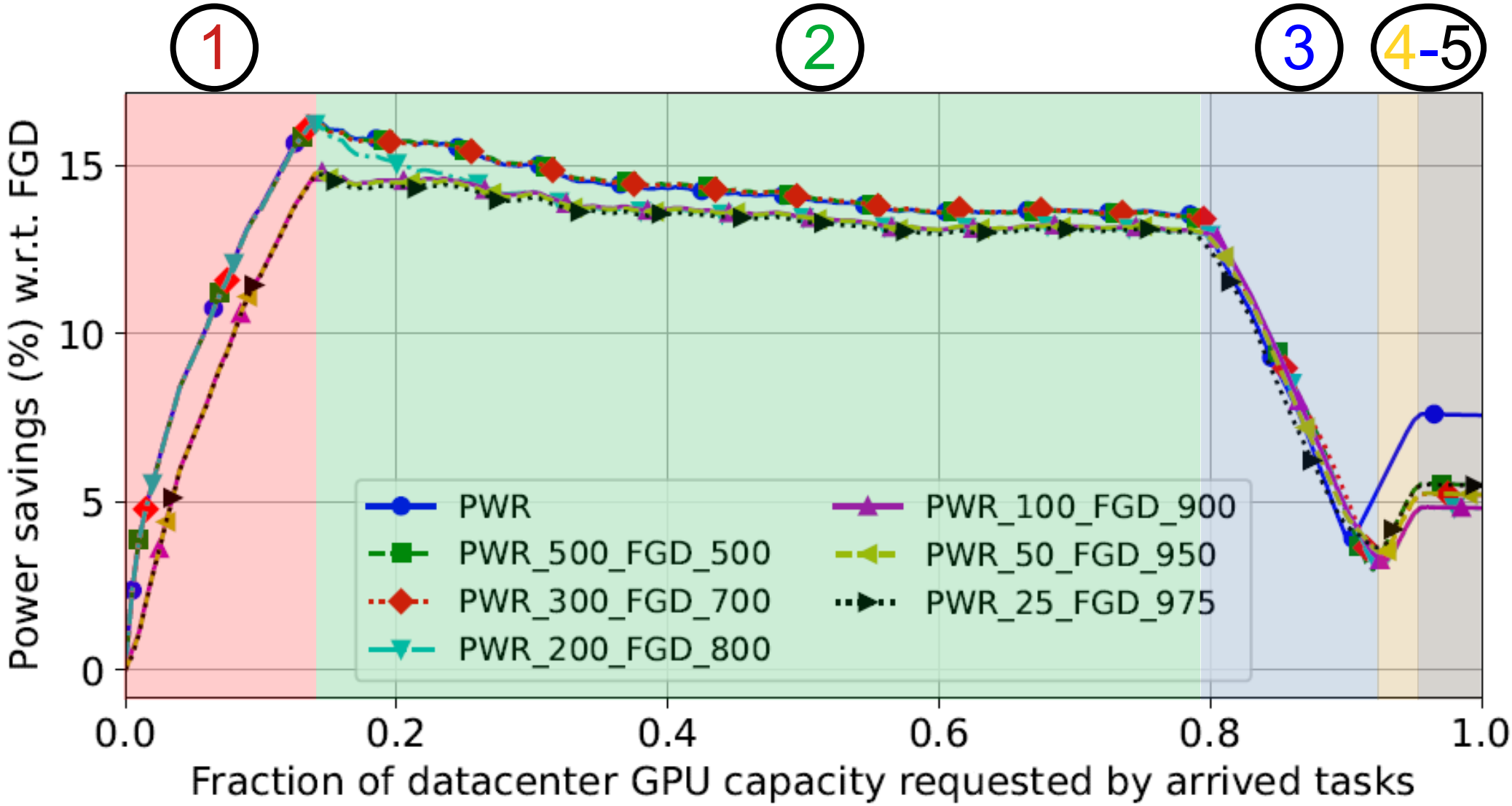
- Still **Default** trace**. Focus** on **power savings vs FGD.** Competitors roughly divide in **two groups**. Consistent **savings ~13-16%**.

- All competitors exhibit a **five-phase pattern.**



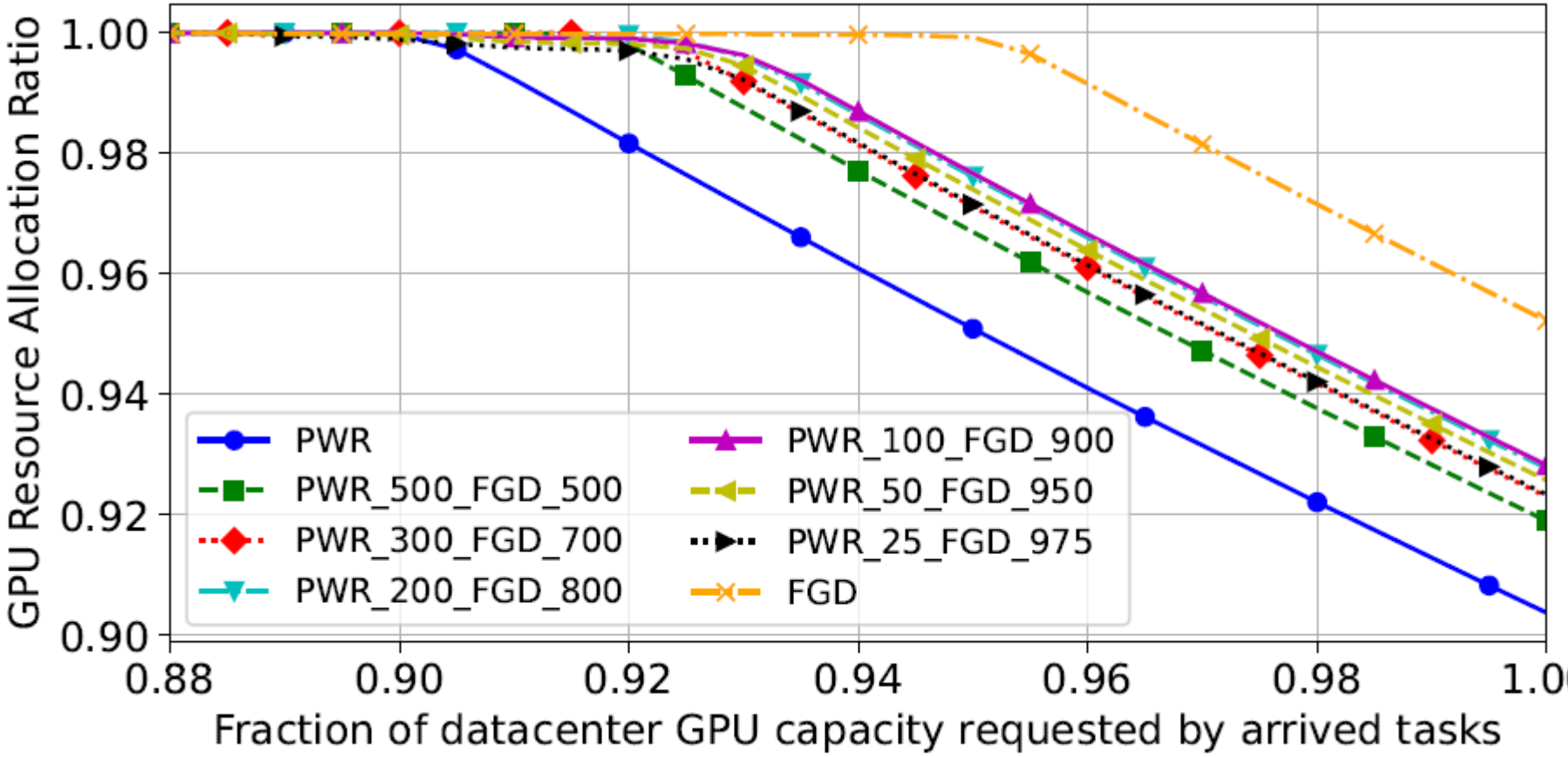
Experimental evaluation: Comparing PWR and PWR+FGD vs FGD / 2

- The **five-phase pattern**:



Experimental evaluation: Comparing PWR and PWR+FGD vs FGD / 3

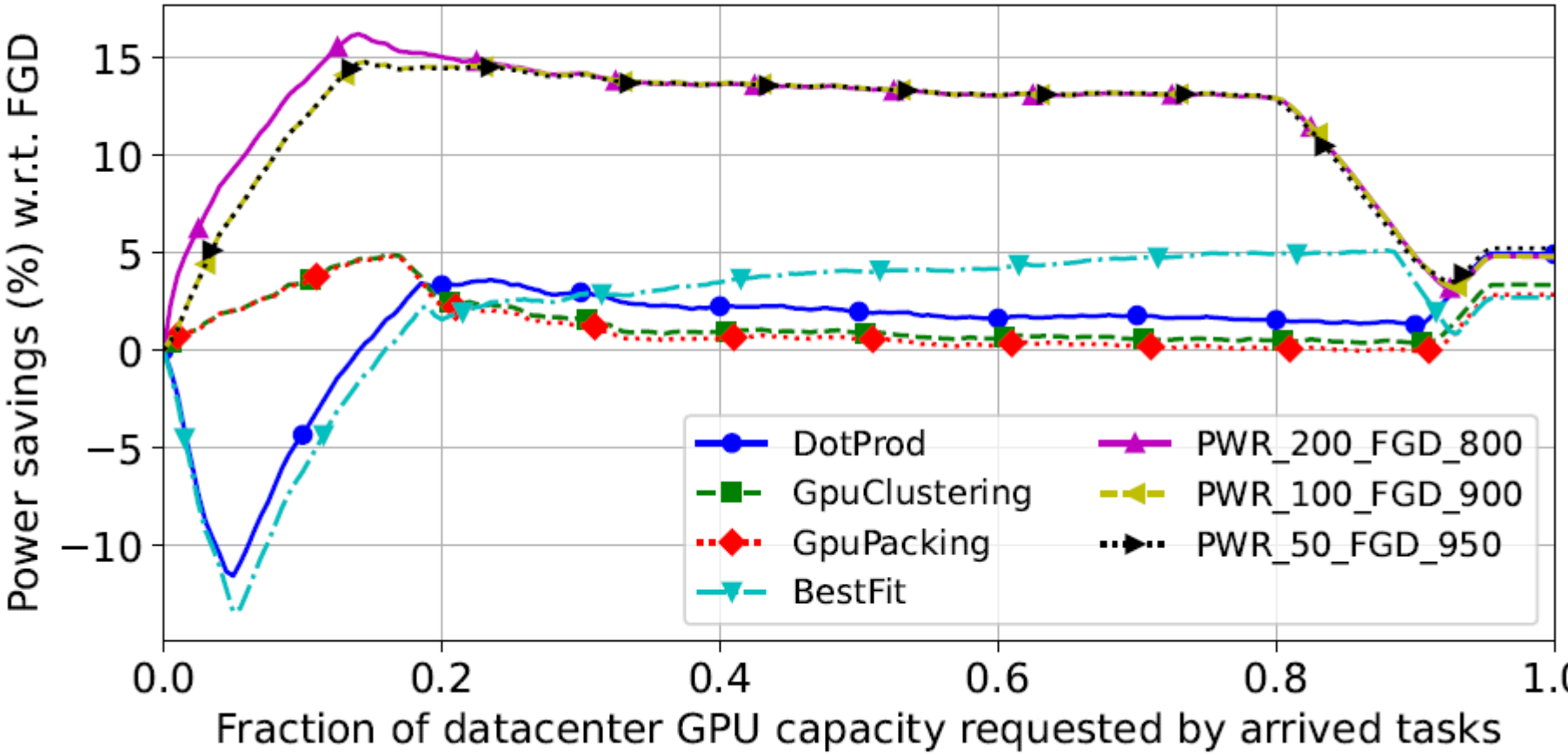
- **GRAR metric**: **FGD** is **best** (expected), as it starts to fail scheduling later than the others.
- **Three PWR+FGD** combinations come reasonably close to FGD: **α ∈ {200, 100, 50}.**



Experimental evaluation: General comparison with Default trace / 1

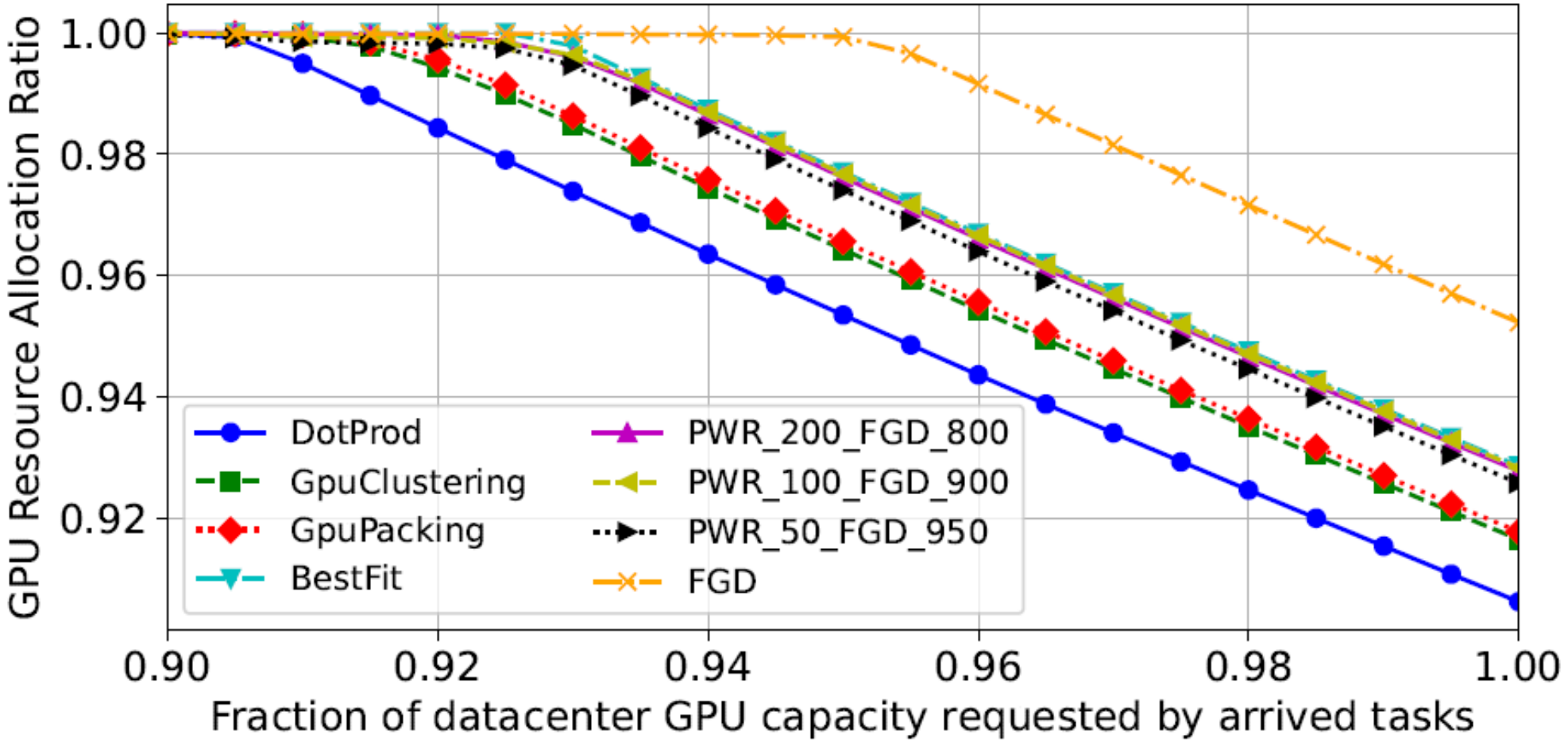
- **Power savings vs FGD**.

- **Substantial gap** between **PWR+FGD combinations** and the **other competitors**.



Experimental evaluation: General comparison with Default trace / 2

- About the **GRAR metric**: **PWR+FGD** are the **second best**, toghether with **BestFit**.



Experimental evaluation: General comparison with Sharing-GPU traces / 1

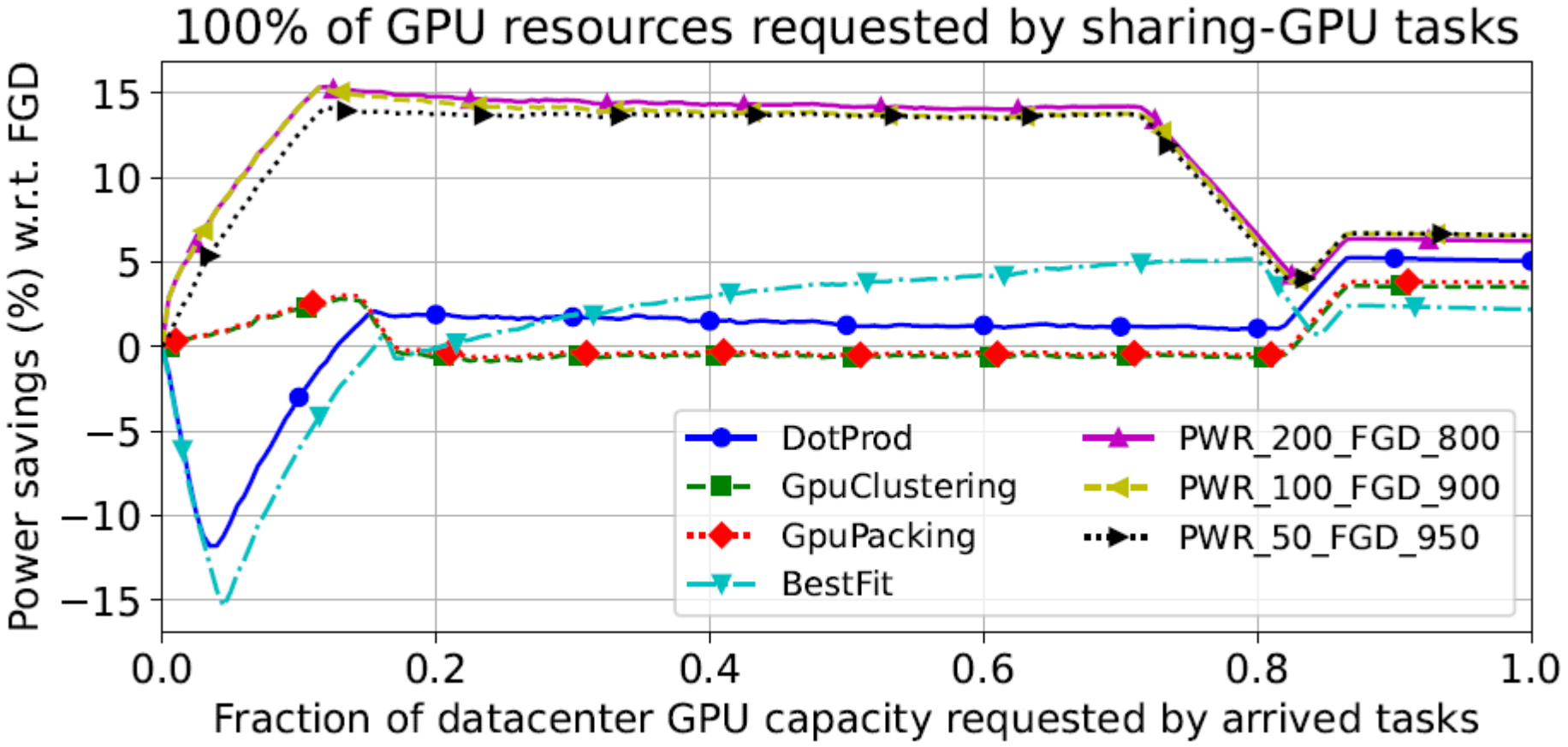
- We **set** the percentage of **GPU resources requested** by **sharing-GPU** tasks at **40%, 60%, 80%,** and **100%** of the **total GPU resources requested** by **GPU tasks**.

- In other words: we **adjust** the **fraction** of **sharing-GPU** and **multi-GPU** tasks, while keeping intra-class distributions fixed.

- **%** of **CPU-only** tasks remains the **same**.

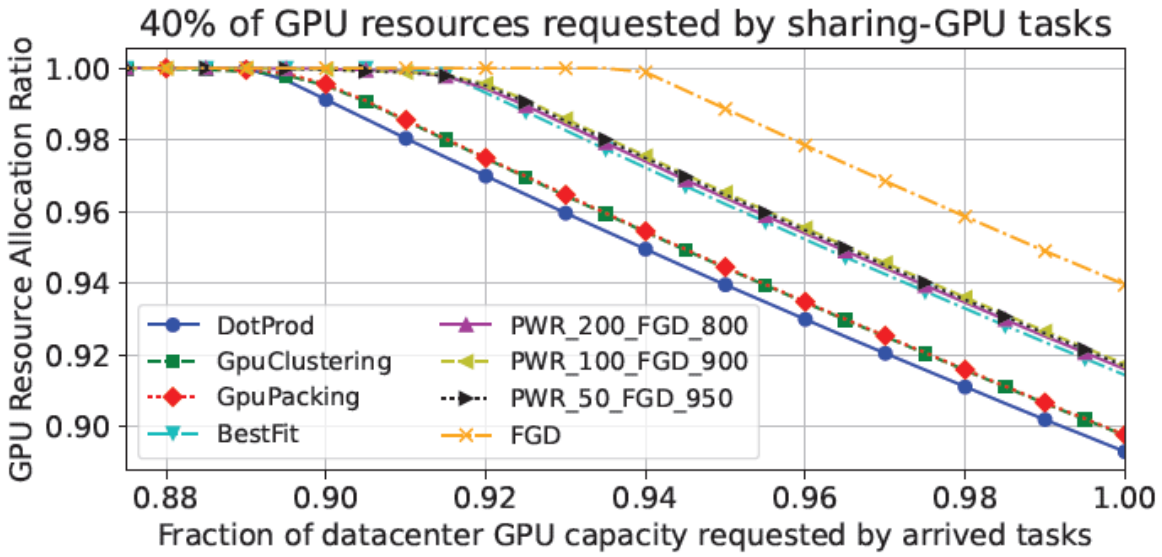
Experimental evaluation: General comparison with Sharing-GPU traces / 2

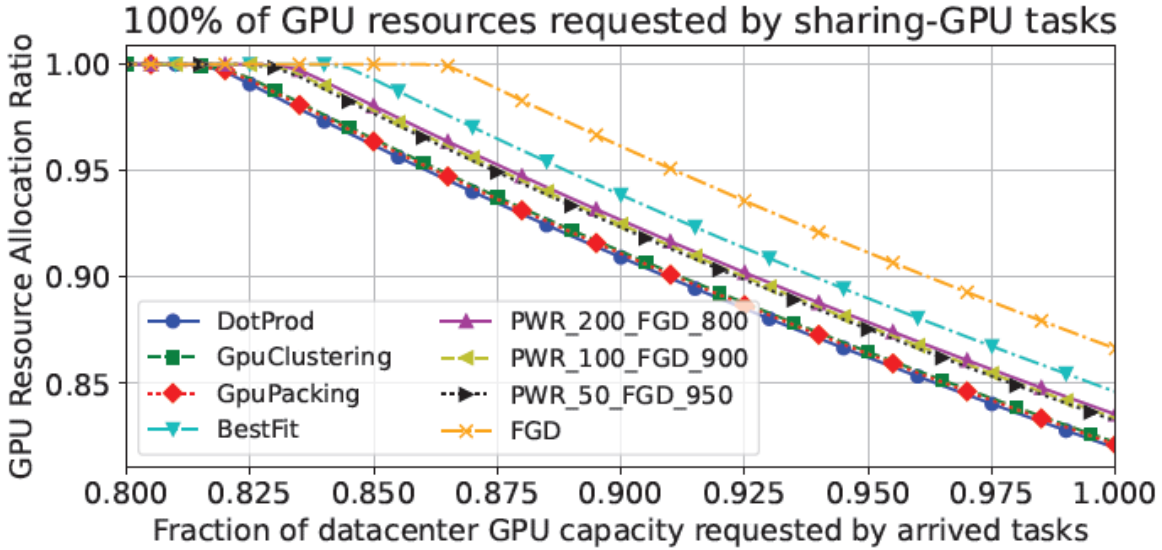
Power savings vs FGD, 100% case -- the other three cases are similar. **Results very similar** to the Default case.



Experimental evaluation: General comparison with Sharing-GPU traces / 3

- **GRAR metric:** 40% and 100% cases.





- All competitors start to **fail earlier** than Default case.
- **Gap** between FGD and competitors **widens** a **little bit**.
- BestFit is second best in the 100% case.

Experimental evaluation: General comparison with Multi-GPU traces / 1

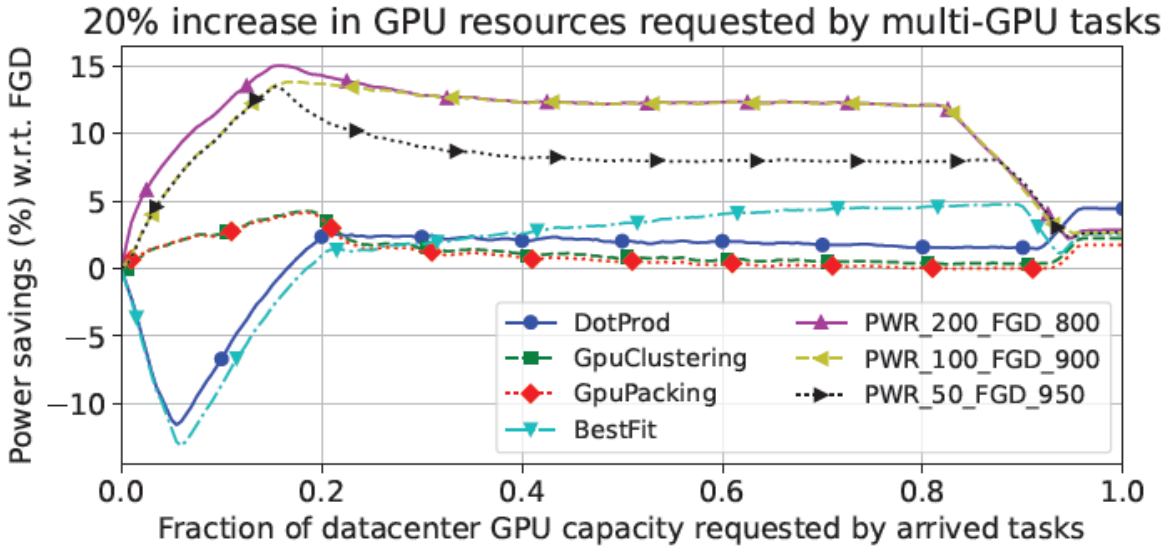
- We **increase** the **amount** of **GPU resources requested** by tasks that **use 1 or more entire GPUs** by **20%, 30%, 40%, and 50%** compared to the **Default** trace.

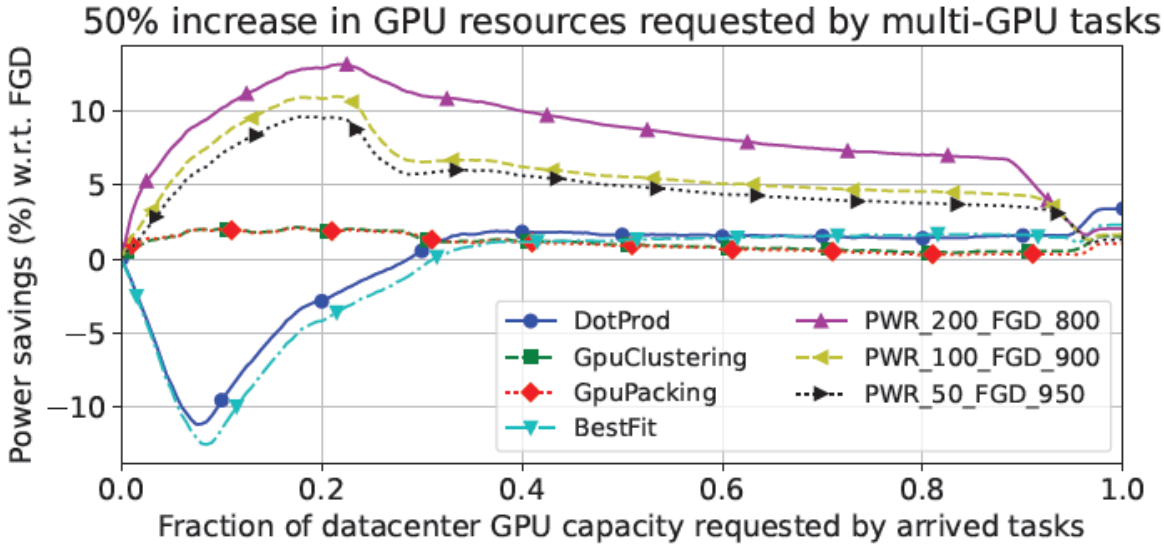
- Done by **increasing** the **total number** of **multi-GPU** tasks while keeping their internal distribution fixed.

- The **numbers** of **no-GPU** and **sharing-GPU** tasks remain **unchanged**.

Experimental evaluation: General comparison with Multi-GPU traces / 2

- **Power savings vs FGD. 20%** and **50% cases.**

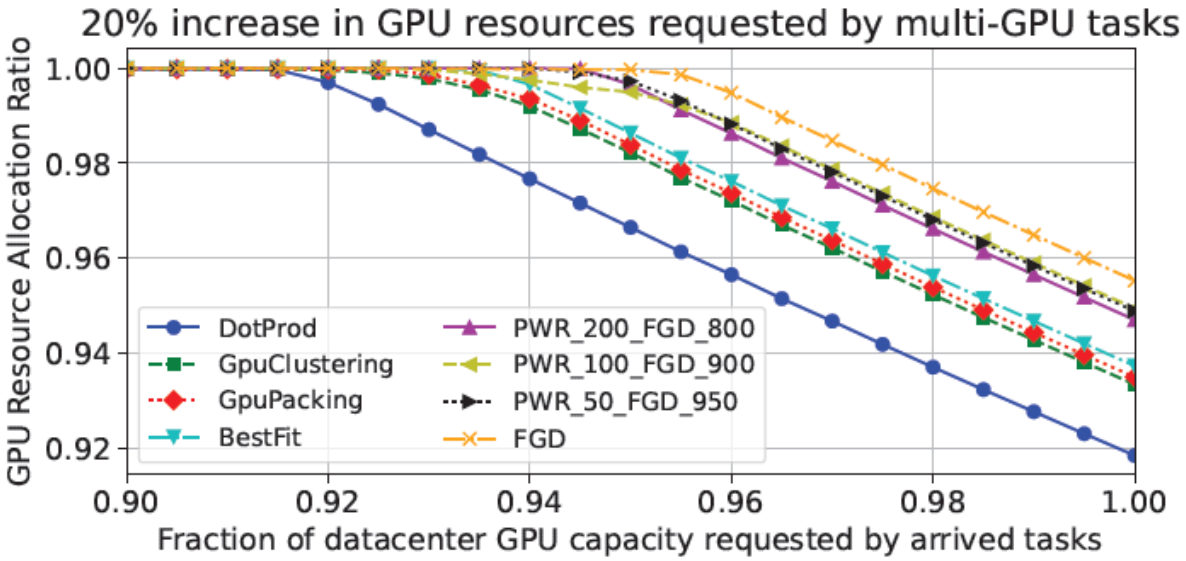


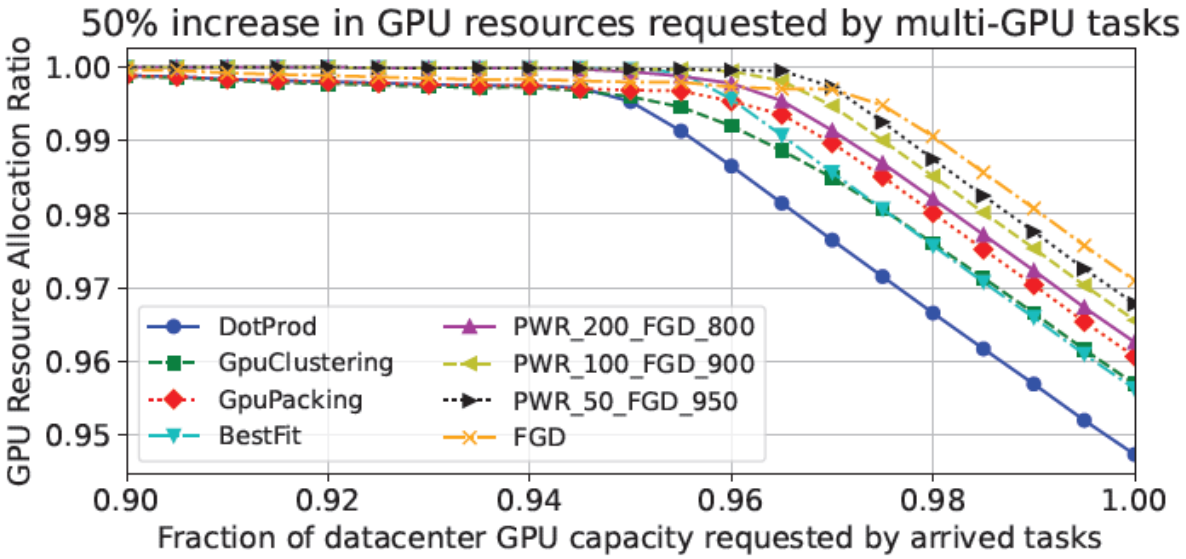


- **Slightly smaller** power savings w.r.t. **Default**.
- **~11-15% (20% case)**
  ~8-13% (50%)
- Smaller, but still substantial, **advantage gap**.

Experimental evaluation: General comparison with Multi-GPU traces / 3

- **GRAR metric. 20%** and **50% cases.**





- PWR+FGD combs. consistently **second best**.
- Also, **closer** to **FGD** than the **Default** case.

Experimental evaluation: General comparison with Constrained-GPU traces / 1

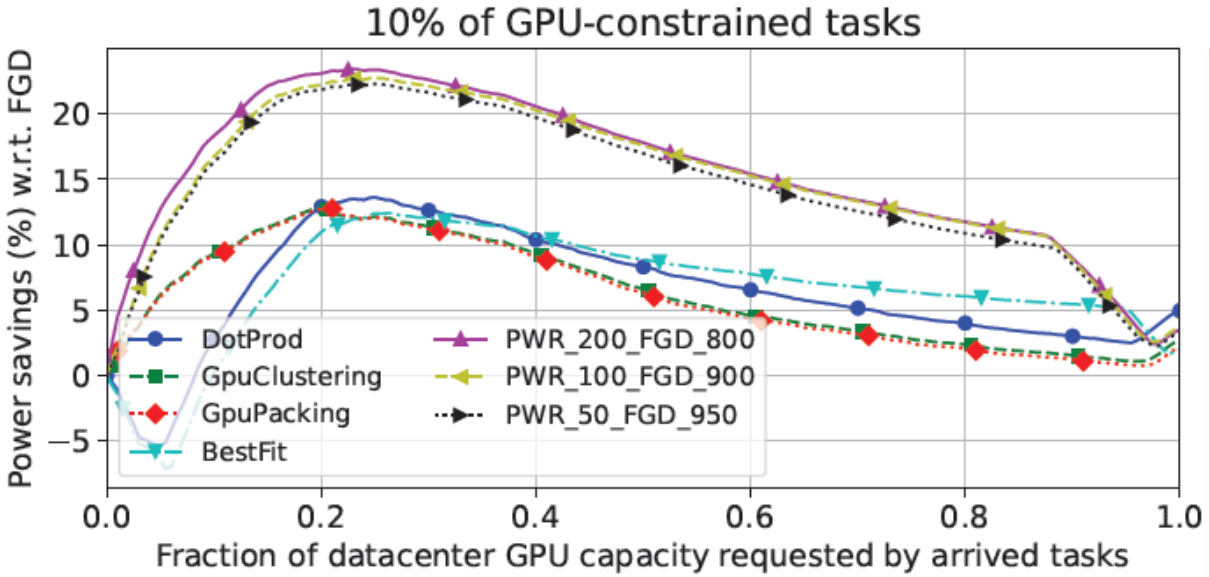
- We set the **percentage** of **GPU tasks** that **request specific GPU models** at **10%, 20%, 25%, and 33%**.

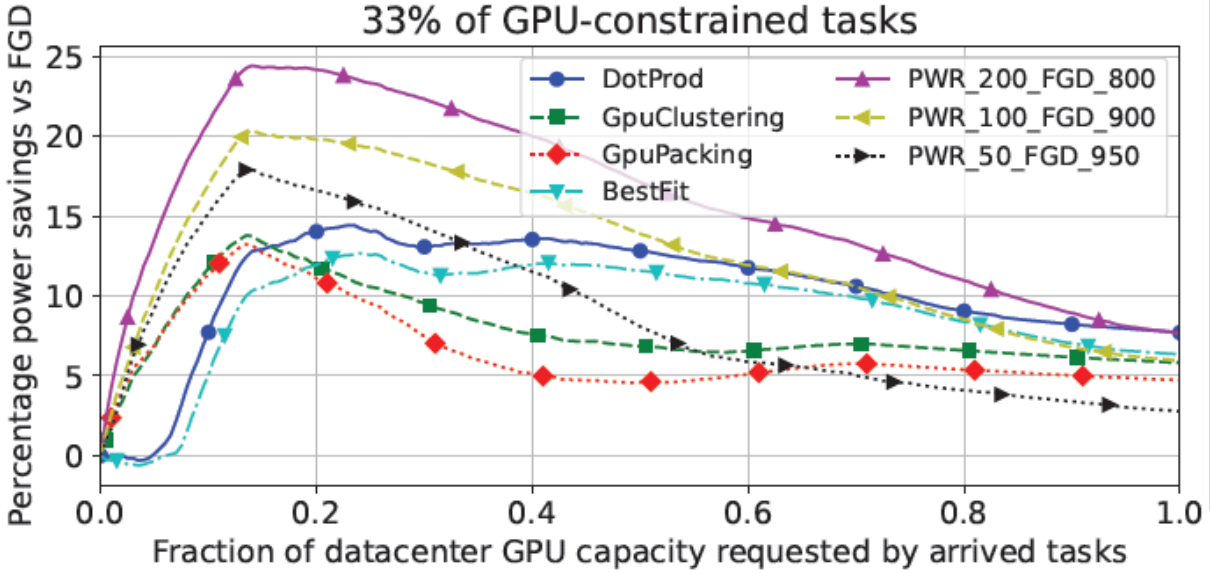
- **All other characteristics match** those of **Default**.

- **Constraints** on **GPUs** put **pressure** on **certain nodes** => we expect **noticeable effects** on the GRAR metric.

Experimental evaluation: General comparison with Constrained-GPU traces / 2

- **Power savings vs FGD. 10%** and **33% cases.**

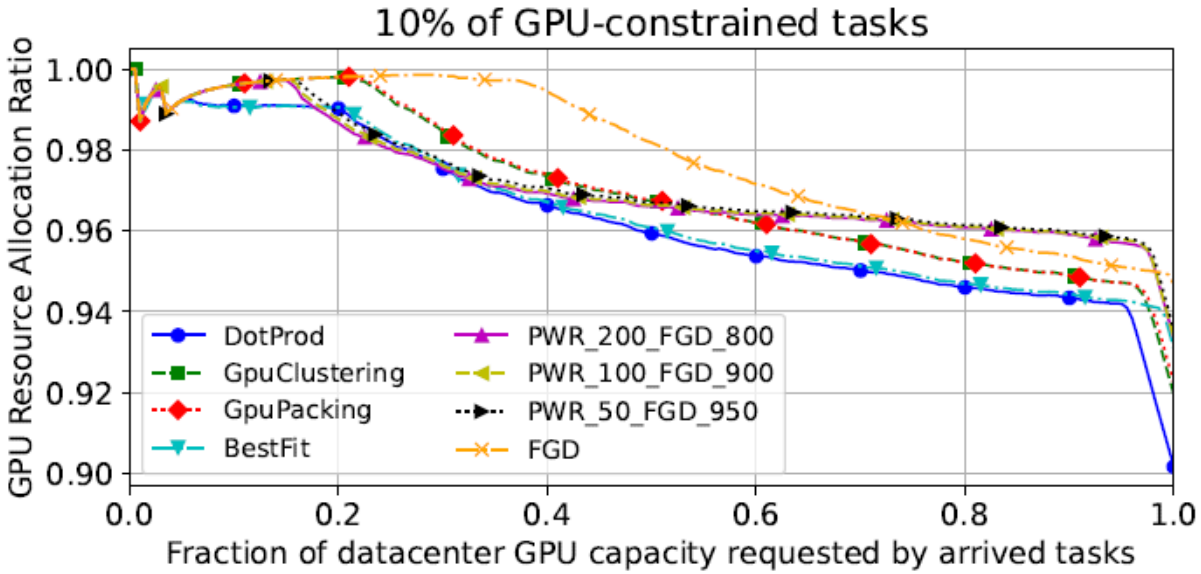


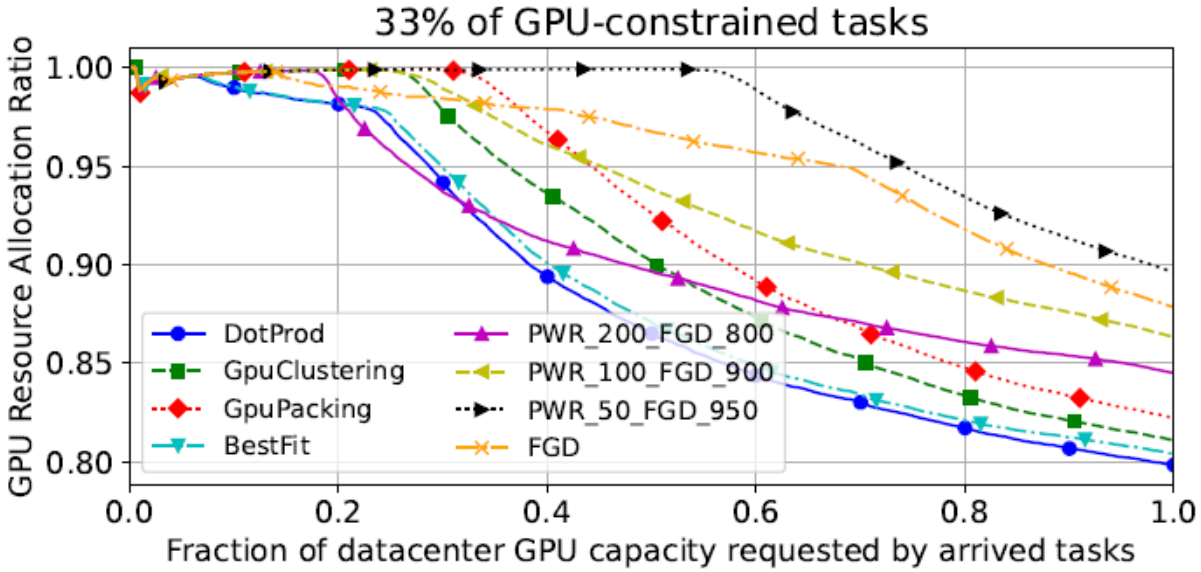


- **Larger** power savings w.r.t. **Default**.
- **~10-24%**.
- As we increase fraction constrained tasks: **advantage gap narrows**.

Experimental evaluation: General comparison with Constrained-GPU traces / 3

- **GRAR metric. 10%** and **33% cases.**





- **Usage pressure** on nodes with specific GPU models!
- Hence, GPU **fragmentation** from the **start**.
- **Our combs.** are **very close** or outperform **FGD**.

Final considerations

- When PWR is properly combined with FGD: overall **scheduling policy** that **effectively reduces power consumption AND GPU fragmentation**.

- Future lines of research…
  - **refine** the **power** consumption **model**;
  - **Use** the **notion** of **target workload** to compute a **node's expected power increase** when scheduling tasks;
  - **Dynamic α** when combining PWR+FGD;
  - Consider the **temporal dimension**.

Thank you!
Questions?

(Our GitHub repository with lots of material!)