On the Scalability of Data Reduction Techniques in Current and Upcoming HPC Systems from an Application Perspective

A. Huebl[1,2], **R. Widera**[1], **F. Schmitt**[2,3], **A. Matthes**[1,2],
N. Podhorszki[4], **J.Y. Choi**[4], **S. Klasky**[4] **and M. Bussmann**[1]

[1] Helmholtz-Zentrum Dresden - Rossendorf

[2] Technische Universität Dresden

[3] NVIDIA ARC GmbH

[4] Oak Ridge National Laboratory

DRBSD-1 Workshop at ISC, Frankfurt *June 22th, 2017*

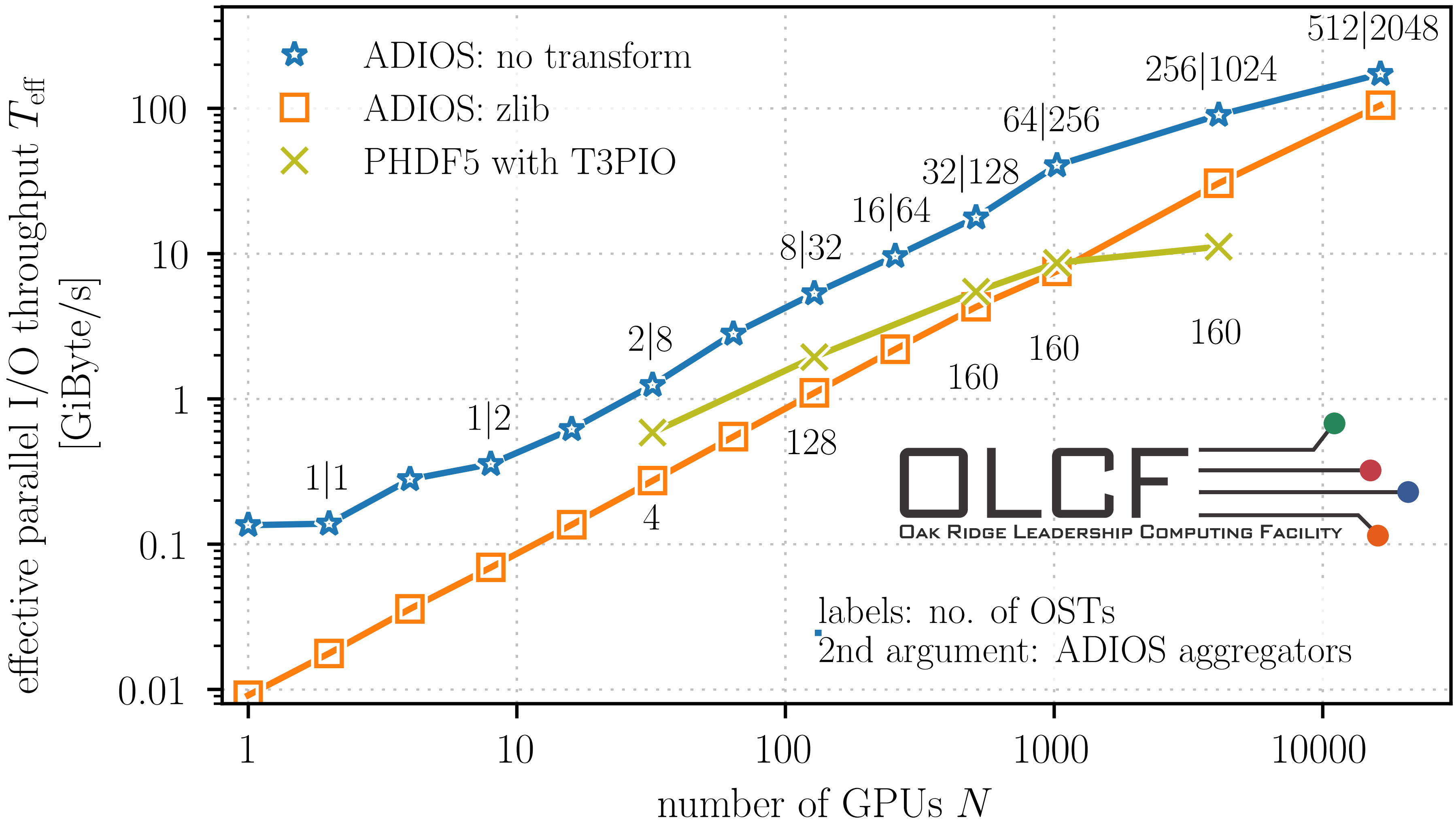
PIConGPU

- fully GPU-parallelized
- scales to full size Titan
- up to 60 GByte/s / node

M. Bussmann et al., SC'13, DOI:10.1145/2503210.2504564

Titan I/O Weak Scaling

$$T_{\text{eff}} \equiv \frac{N \times S}{t_{\text{I/O}}}$$



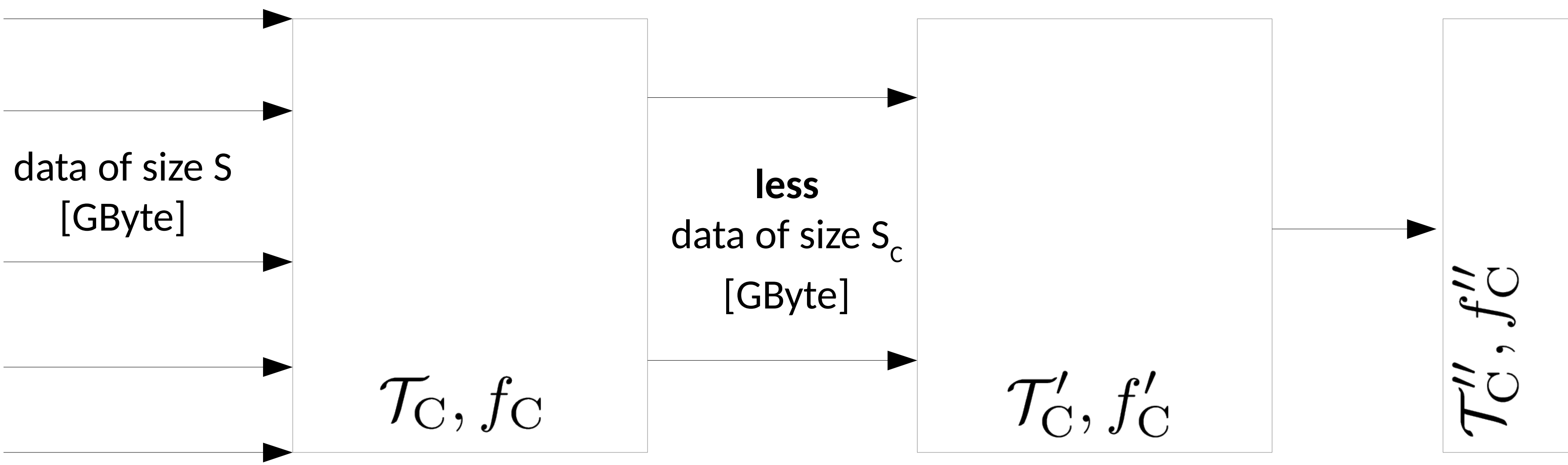
- Isn't data reduction always for free?
- Modeling of Parallel I/O
- Trading Compute Resources for I/O

Characterization Of Data Reduction

Data Reduction

f_C reduction (compression) ratio S_C / S

$\mathcal{T}_C$ reduction (compression) throughput T_C / T_{memcpy}

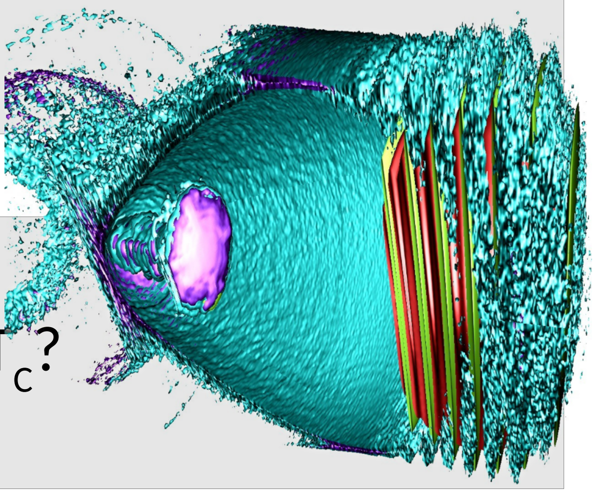


Data Reduction Examples

Binning of a spectrogram f_C->0, T_C~T_{memcpy}

cp to NVRAM f_C=1, T_C<T_{memcpy}

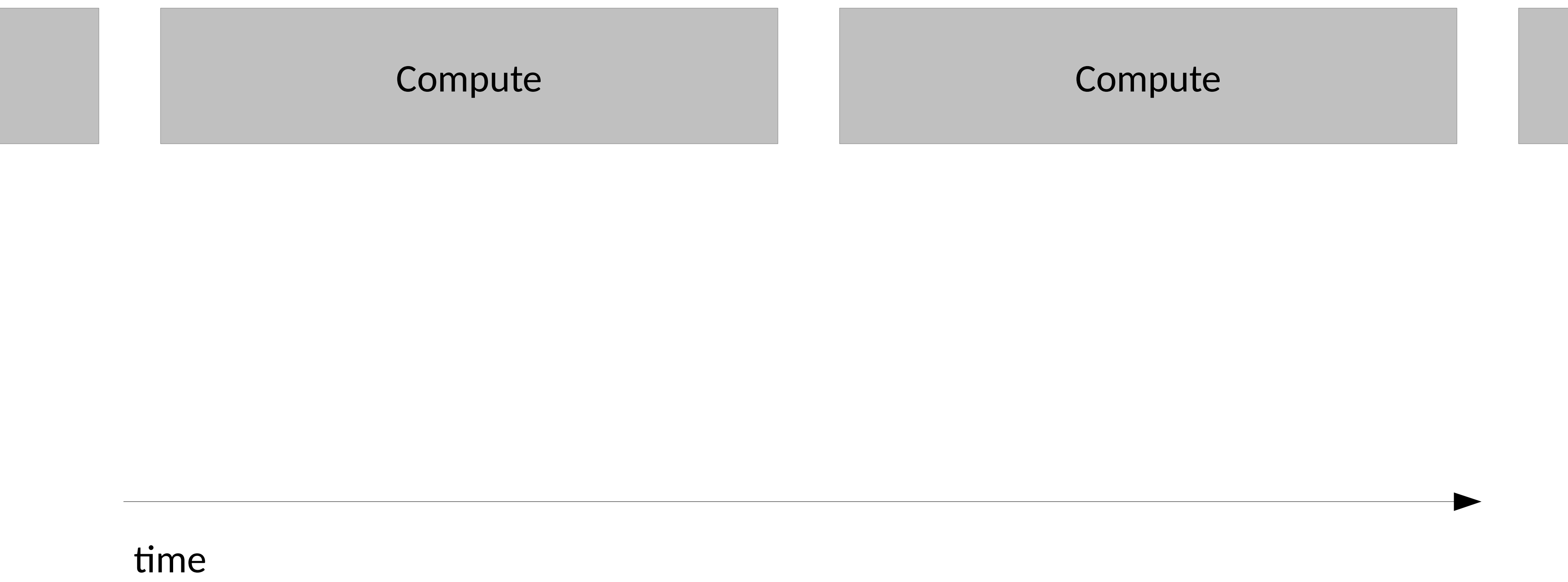
Ray-cast or photo-realistic ray-trace f_C->0, T_C?



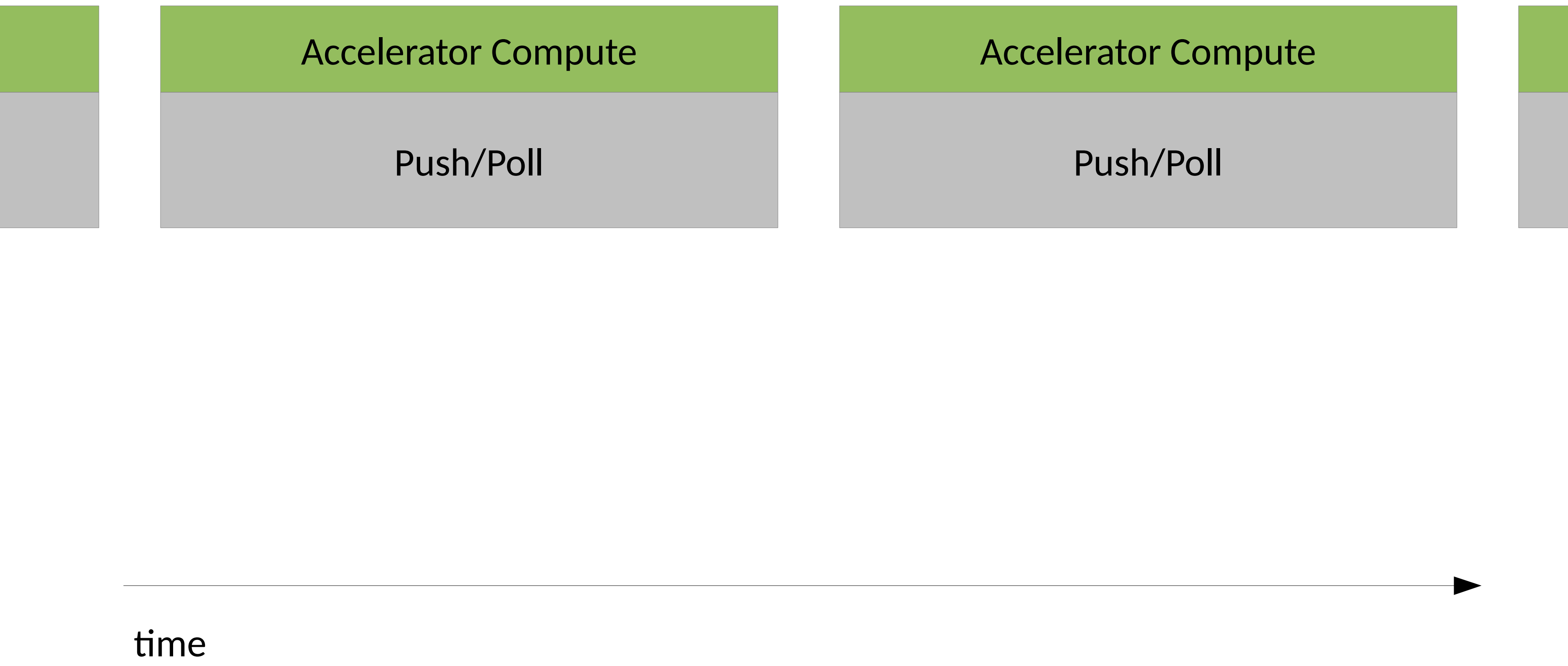
A. Matthes, A. Huebl et al., ISC 2017, DOI:10.14529/jsfi160403

Modeling Real-World Data Flows

Application View

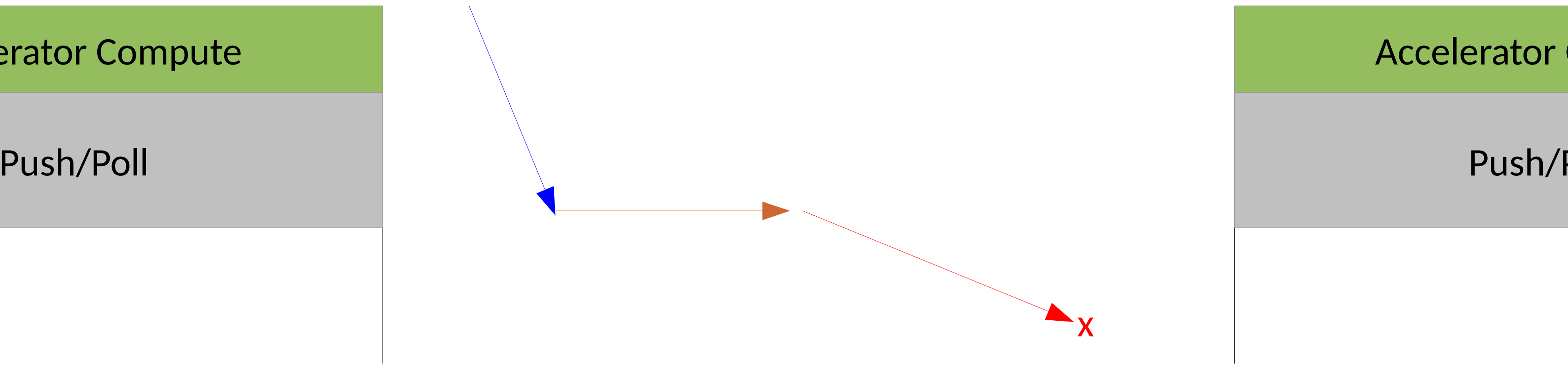


Application View



Application View

Unreduced, Synchronous I/O



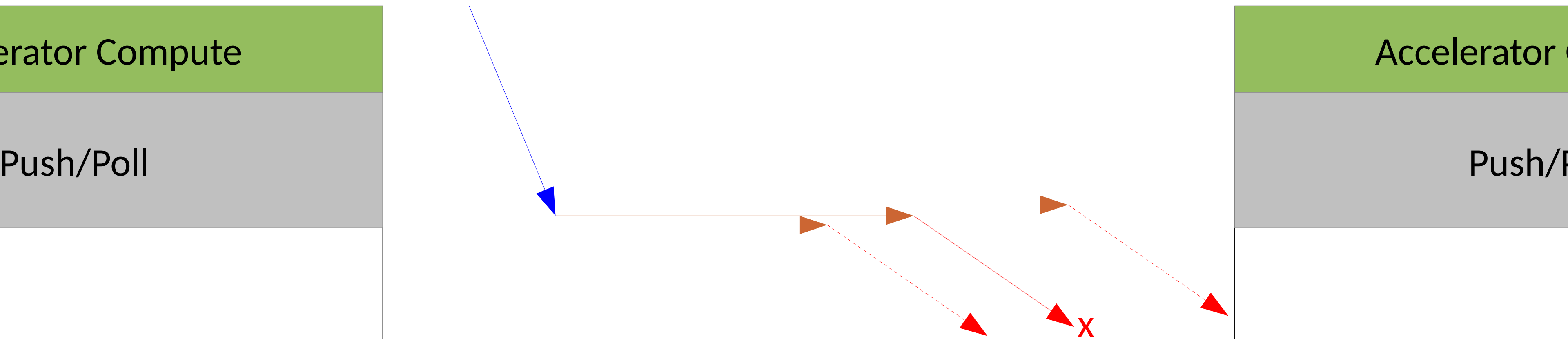
$$t_{\mathrm{I/O}} \equiv t_{\mathrm{prep}} + t_{\mathrm{memcpy}} + t_{\mathrm{off\ RAM}}$$

time

$$+ \frac{S}{T_{\mathrm{memcpy}}} + \frac{S}{T_{\mathrm{FS}}}$$

Application View

<u>Reduced</u>, Synchronous I/O



$$t_{\mathrm{I/O}} \equiv t_{\mathrm{prep}} + \frac{S}{T_C} + \frac{f_C \times S}{T_{\mathrm{FS}}}$$

time

Comparison Synchronous I/O

Unreduced

$$t_{\mathrm{I/O}} \equiv t_{\mathrm{prep}} + \frac{S}{T_{\mathrm{memcpy}}} + \frac{S}{T_{\mathrm{FS}}}$$

$\updownarrow$ > **when?**

Reduced

$$t_{\mathrm{I/O}} \equiv t_{\mathrm{prep}} + \frac{S}{T_C} + \frac{f_C \times S}{T_{\mathrm{FS}}}$$

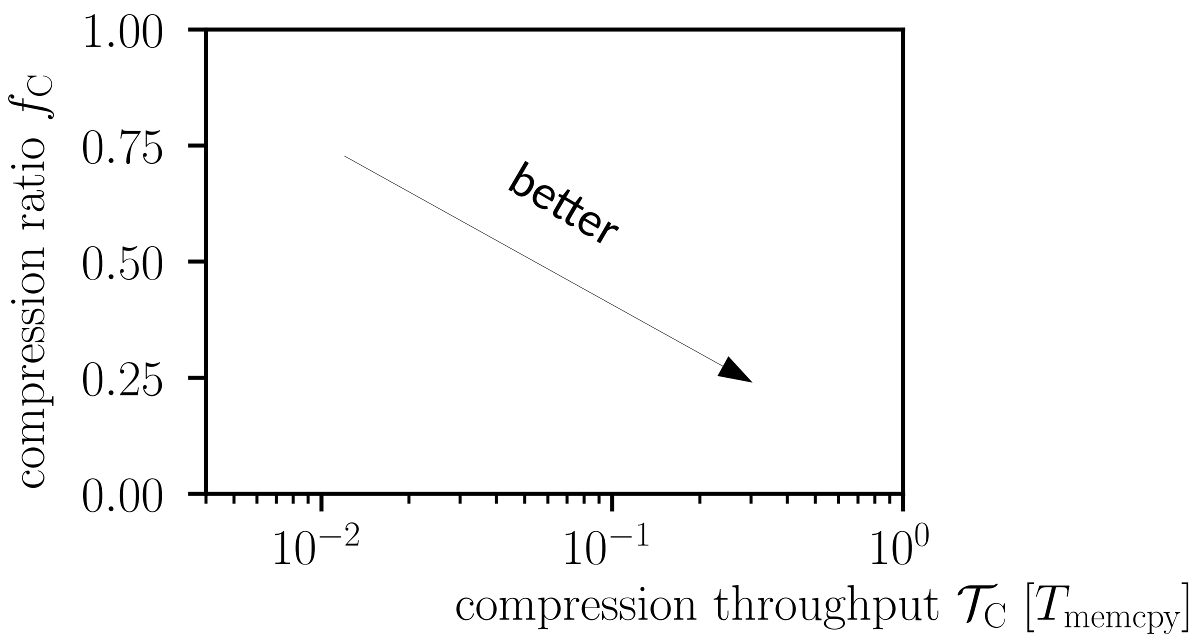
Break-Even Threshold

$$\frac{\mathcal{T}_\mathrm{C} \times (1 - f_\mathrm{C})}{1 - \mathcal{T}_\mathrm{C}} > \mathcal{T}_\mathrm{FS}$$

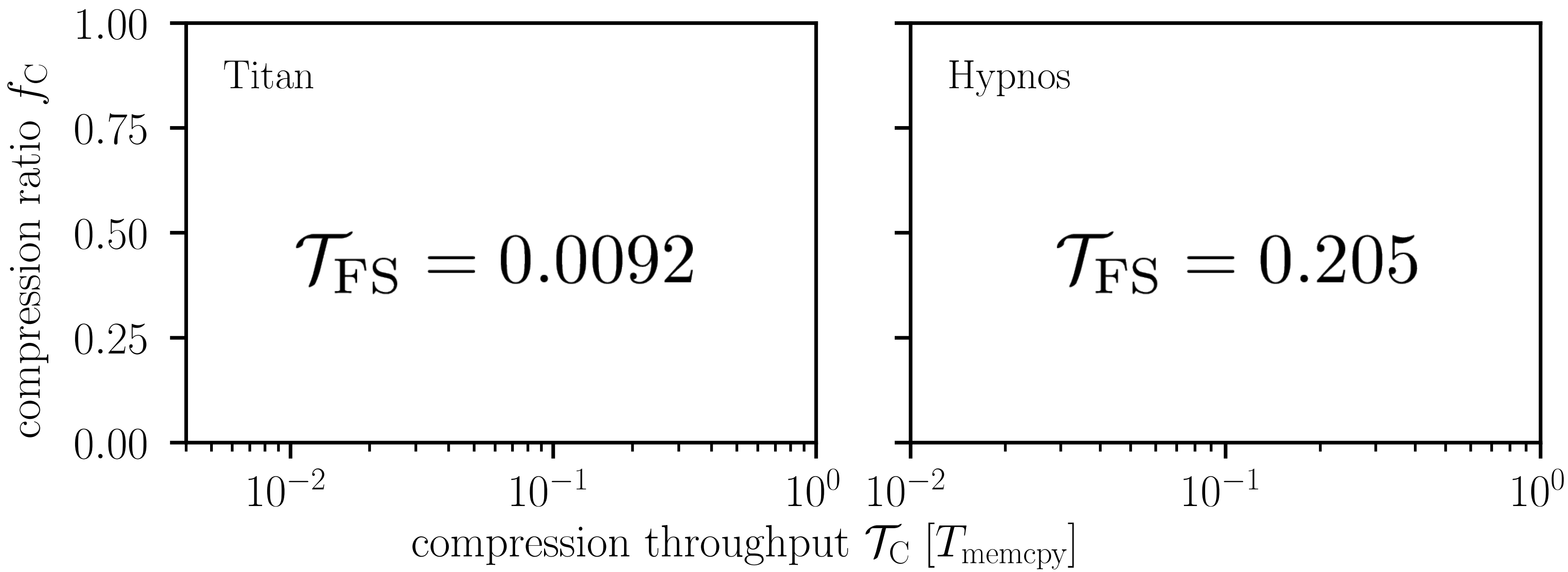
$$f_\mathrm{C} \equiv \frac{S_\mathrm{C}}{S} \quad \mathcal{T}_\mathrm{C} \equiv \frac{T_\mathrm{C}}{T_\mathrm{memcpy}} \quad \mathcal{T}_\mathrm{FS} \equiv \frac{T_\mathrm{FS}}{T_\mathrm{memcpy}}$$

Compression
(as a data reduction example)

Ex-Situ Measurements for Compression Algorithms

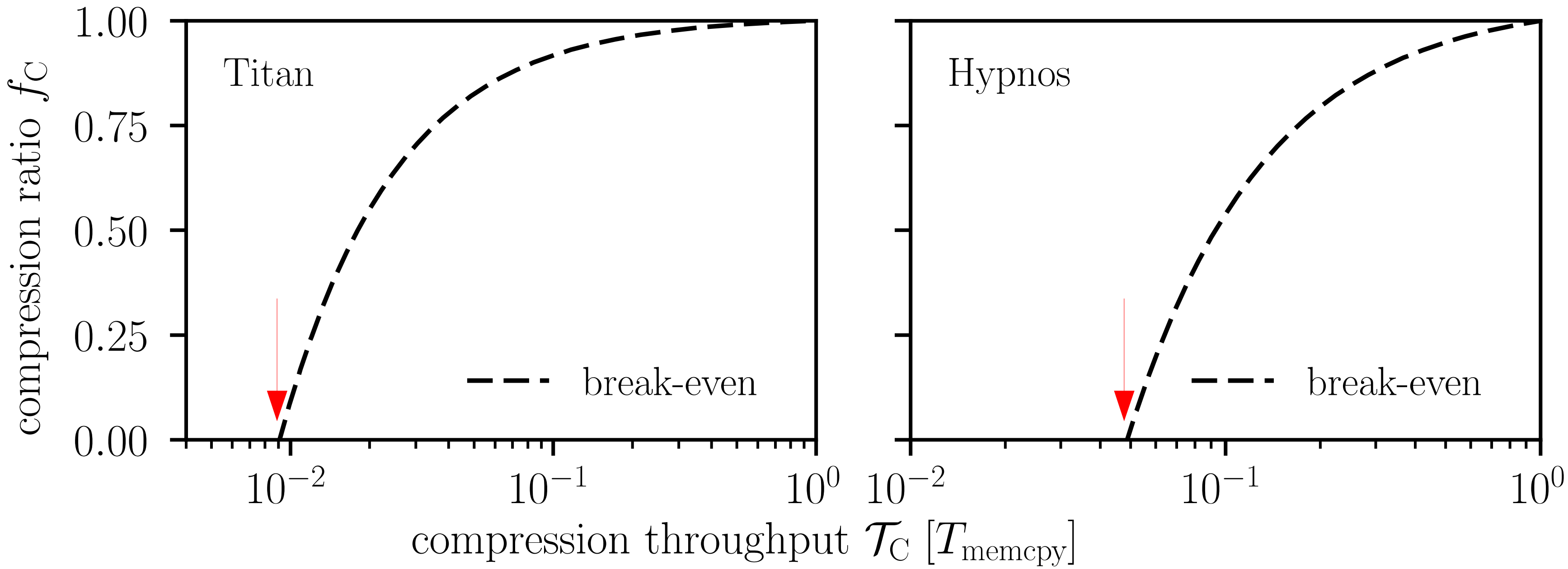


Ex-Situ Measurements for Compression Algorithms

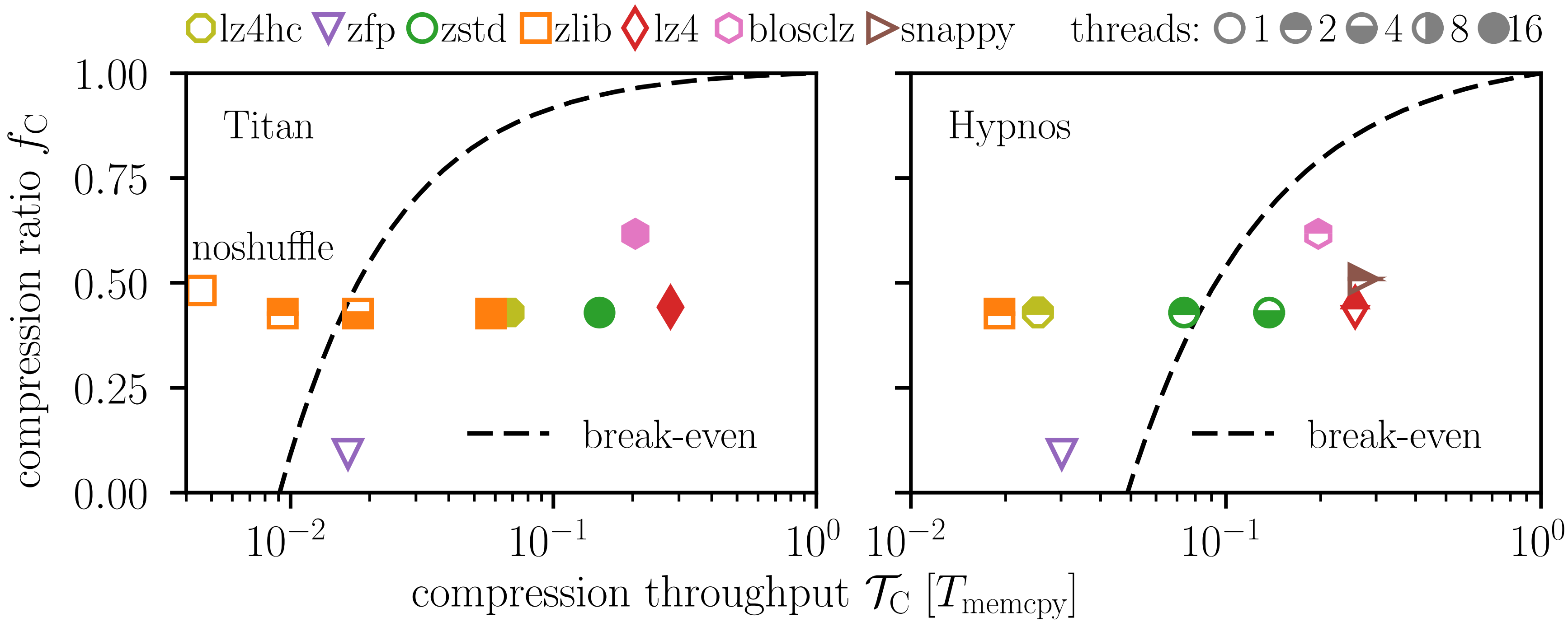


Ex-Situ Measurements for Compression Algorithms

$$\mathcal{T}_\mathrm{C} > \frac{\mathcal{T}_\mathrm{FS}}{\mathcal{T}_\mathrm{FS} + 1}$$



Ex-Situ Measurements for Compression Algorithms



Zfp 0.5.1: three uncompressed bits / scalar; on *particle data*

Blosc 1.11.4-dev: add bitshuffle pre-conditioner

Impact on Applications

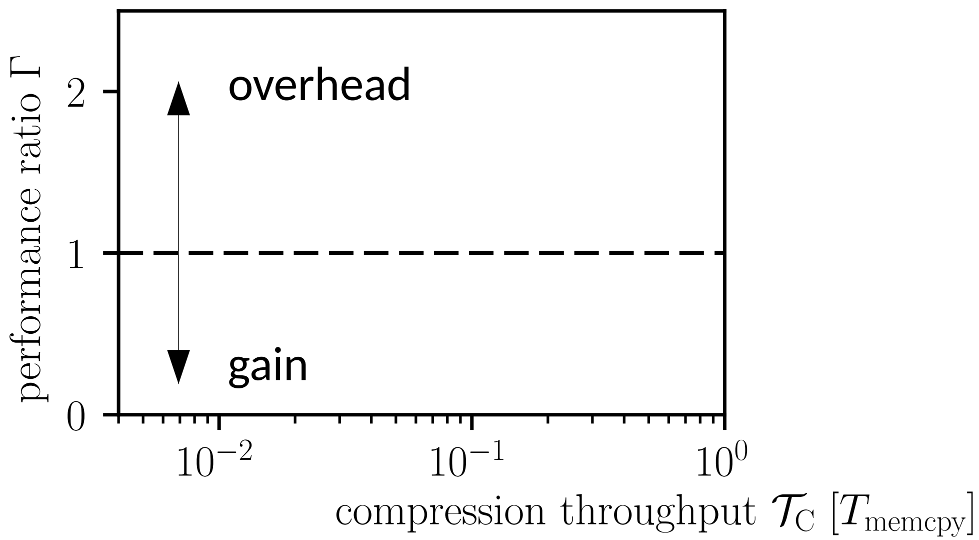
Predicting Feasible Reduction Algorithms to Reduce I/O Time

Summary of System Characteristics

f_C	reduction (compression) ratio	[0:1]
$\mathcal{T}_\mathrm{C}$	reduction (compression) throughput	[$\mathrm{T}_\mathrm{memcpy}$]
$\mathcal{T}_\mathrm{FS}$	write to next stage (e.g. FS)	[$\mathrm{T}_\mathrm{memcpy}$]
t_prep	preparation for the reduction stage	(seconds)

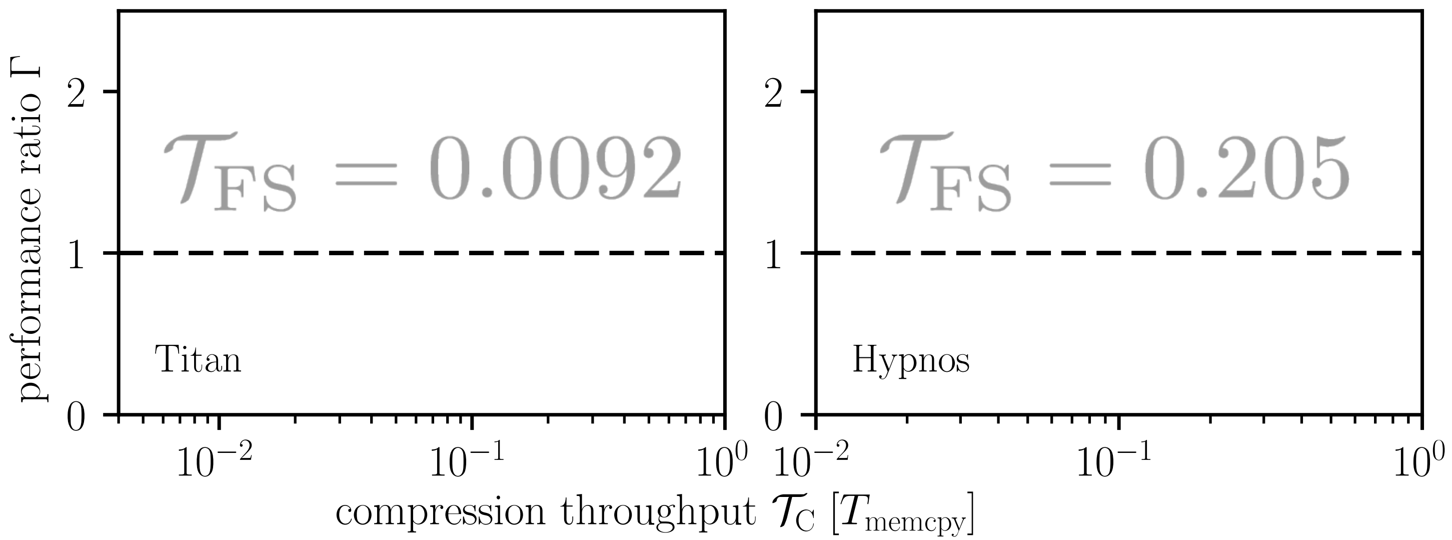
Application Performance: Gain or Loss?

$$\Gamma \equiv \frac{t_{\mathrm{I/O}}^{\mathrm{reduced}}}{t_{\mathrm{I/O}}^{\mathrm{unreduced}}}$$



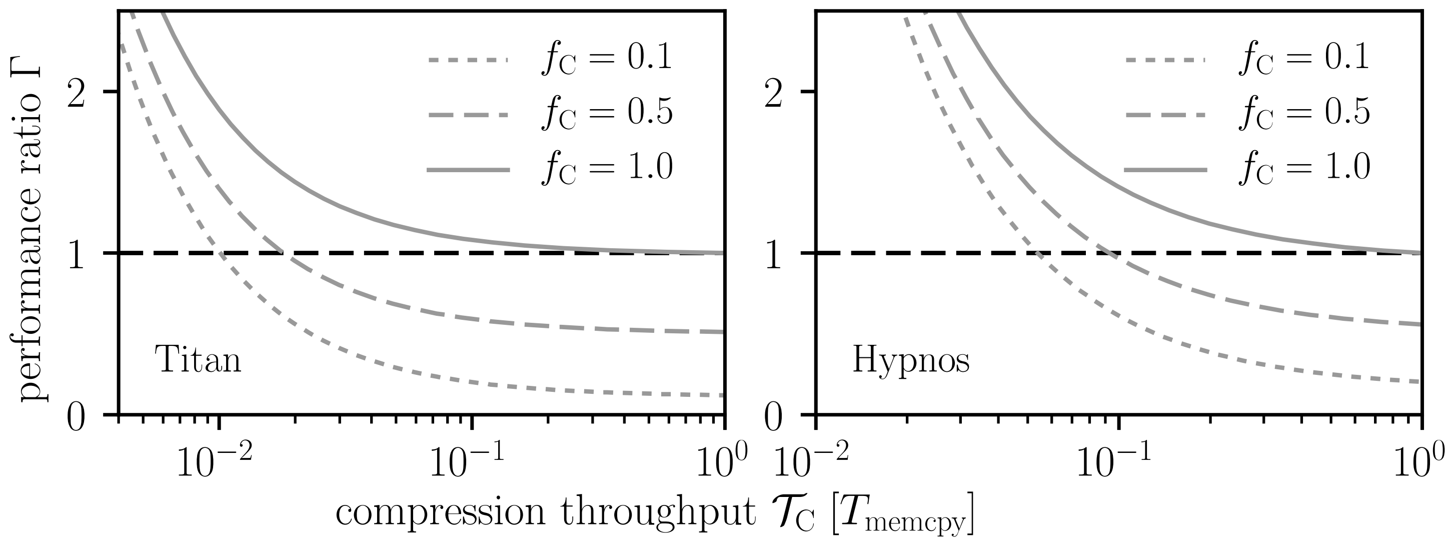
Application Performance: Gain or Loss?

$$\Gamma \equiv \frac{t^{\text{reduced}}_{\text{I/O}}}{t^{\text{unreduced}}_{\text{I/O}}}$$



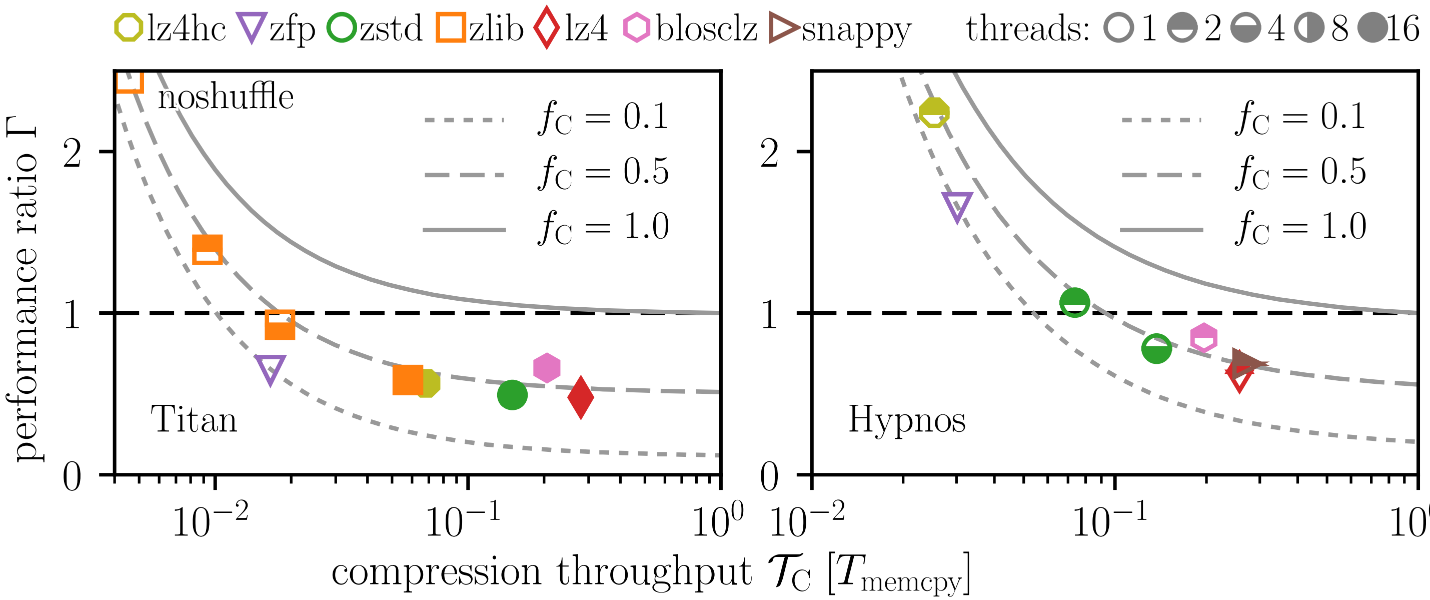
Application Performance: Gain or Loss?

$$\Gamma \equiv \frac{t_{\mathrm{I/O}}^{\mathrm{reduced}}}{t_{\mathrm{I/O}}^{\mathrm{unreduced}}}$$



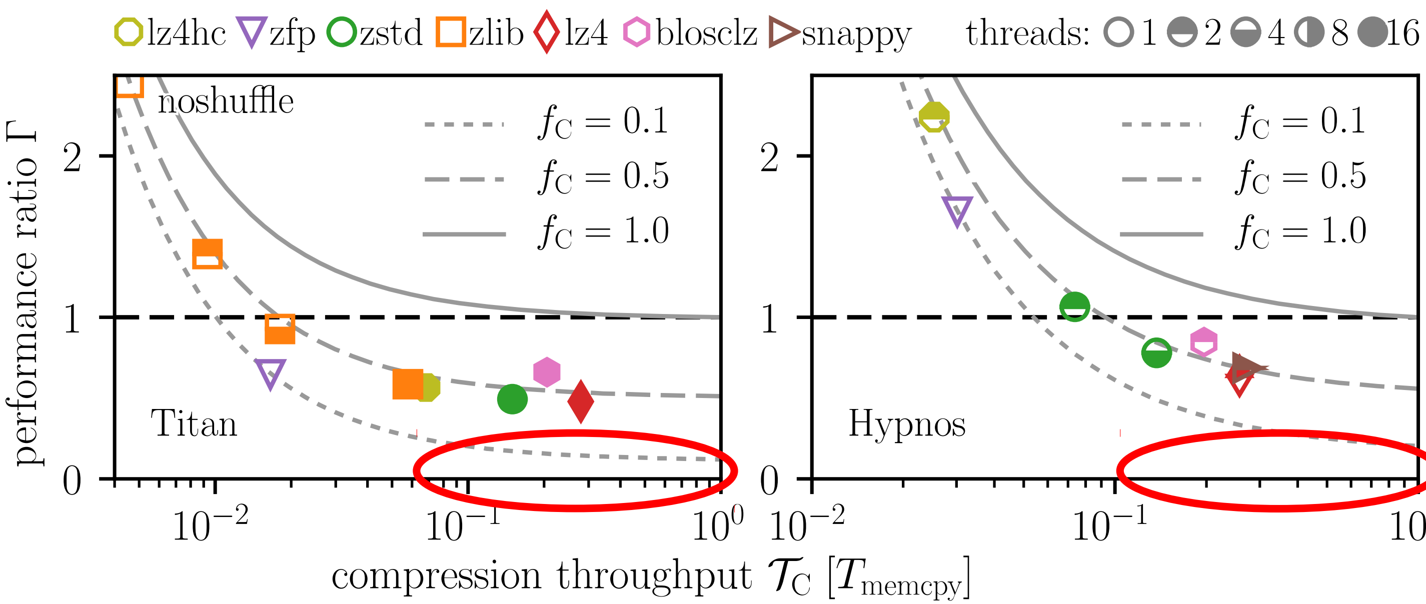
Application Performance: Gain or Loss?

$$\Gamma \equiv \frac{t_{\mathrm{I/O}}^{\mathrm{reduced}}}{t_{\mathrm{I/O}}^{\mathrm{unreduced}}}$$



Application Performance: Gain or Loss?

$$\Gamma \equiv \frac{t_{\mathrm{I/O}}^{\mathrm{reduced}}}{t_{\mathrm{I/O}}^{\mathrm{unreduced}}}$$

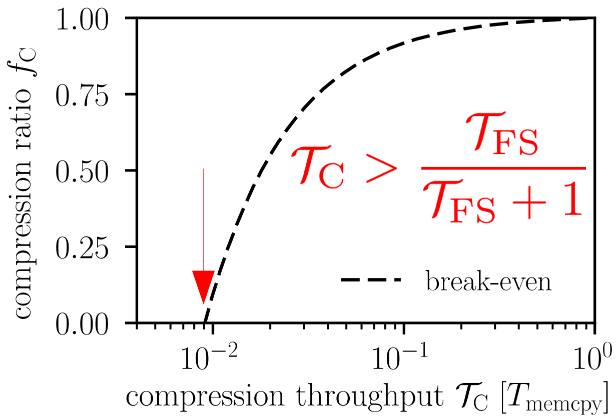


Enable application I/O on upcoming systems

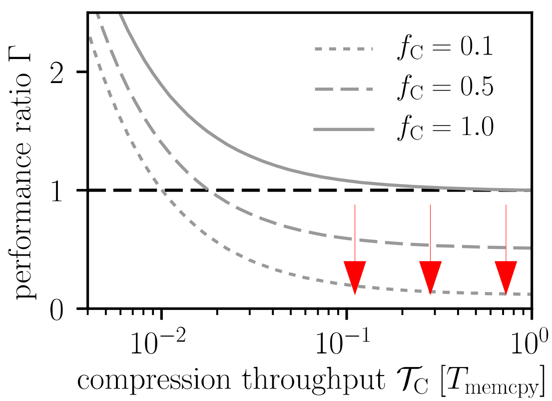
via $\Gamma \ll 1$ (Γ=1 will be expensive)

Consequences for Reduction Techniques

Throughput first



Aggressive Reduction next



Copy to burst buffers immediately "asynchronous I/O"

interleave computation with: t_{reduce}, $t_{\mathrm{off\ RAM}}$

still: backlog on subsequent I/O

Users need <u>easily programmable</u> **I/O stages**

… & *each stage* bound by the above (else: backlog)

Even with data reduction, there is no free lunch...

... unless you are *fast* $\mathcal{T}_\mathrm{C}$

... only then take a big bite f_C

Thank you for your attention!

This research used resources of the Oak Ridge Leadership Computing Facility located in the Oak Ridge National Laboratory, which is supported by the Office of Science of the Department of Energy under Contract DE-AC05-00OR22725.

This project has received funding from the European Unions Horizon 2020 research and innovation programme under grant agreement No 654220.

Backup Slides
(Scenarios of Interest)

On-GPU Compression via Copy then → NIC

$$t_{\mathrm{I/O}} \equiv {\color{blue}t_{\mathrm{prep}}} {\color{orange}+ t_{\mathrm{reduce}}} {\color{red}+ f_C \times t_{\mathrm{off\ RAM}}}$$

\- -- ==

needs free global GPU RAM

(GPU apps are greedy, better have $f_c \rightarrow 0$)

Who compresses? User or I/O library?

Persistently In-GPU Compressed → direct to NIC

$$t_{\mathrm{I/O}} \equiv \underbrace{\cancel{t_{\mathrm{prep}}}}_{\sim 0} + \underbrace{\cancel{t_{\mathrm{reduce}}}}_{\sim 0} + \underbrace{f_C \times t_{\mathrm{off\ RAM}}}_{==}$$

needs algorithmic & data structure support

Every access costs:

- Flop/s
- registers

Examples exists: e.g. zfp

Titan vs. Summit Prediction

$$N_{\text{nodes}} T_{\text{FS}} : \times\, 2.5^{\text{faster}} \times 4^{\text{less nodes}}$$

$$S/\text{sec} : \times\, 10^{\text{faster}} \times 4^{\text{"fatter"}}$$

$$\mathcal{T}_{\text{FS}} \approx \times\, 0.25$$

$$\text{I/O} : 4.0 \text{ more costly}$$

$$t_{\text{I/O}} \equiv t_{\text{prep}} + \frac{S}{T_C} + \frac{f_C \times S}{T_{\text{FS}}}$$

== or + **==** **x 4**

- slow reductions hit break-even earlier
- but $t_{I/O}$ (starts Γx4 !) does *not* decrease linear with f_C alone

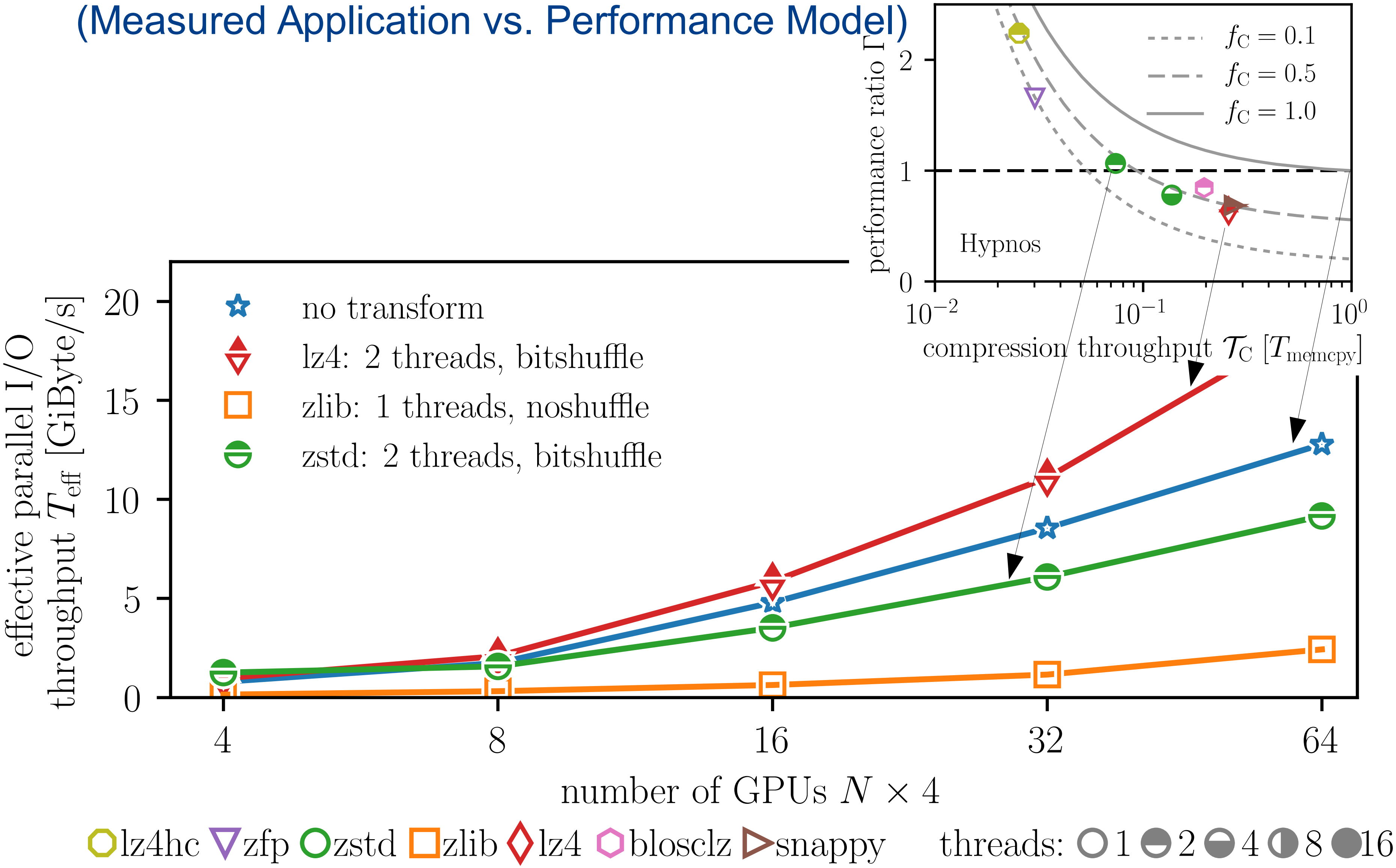
Titan vs. Hypnos

Table 1. PIConGPU I/O benchmark systems, both commissioned in 2012/13: relevant system characteristics and single node average filesystem throughput T_{FS}, defined as the design parallel bandwidth $B_{parallel}$ divided by N nodes

	Titan	Hypnos (queue: 'k20')
GPUs / node	1× K20x	4× K20m
CPUs / node	1× AMD Opteron 6274	2× Intel Xeon E5-2609
CPU-cores / GPU	16 (8 FP)	2
GPU / CPU Flop/s (DP)	9.3 : 1	7.6 : 1
file system	Spider/Lustre	GPFS
$B_{parallel} = T_{FS} * N$ [GiByte/s]	1000	20
T_{FS} [GiByte/s]	0.055	1.25
CPU T_{memcpy} [GiByte/s]	6.0	6.1
maximum number of nodes N_{max}	18000	16

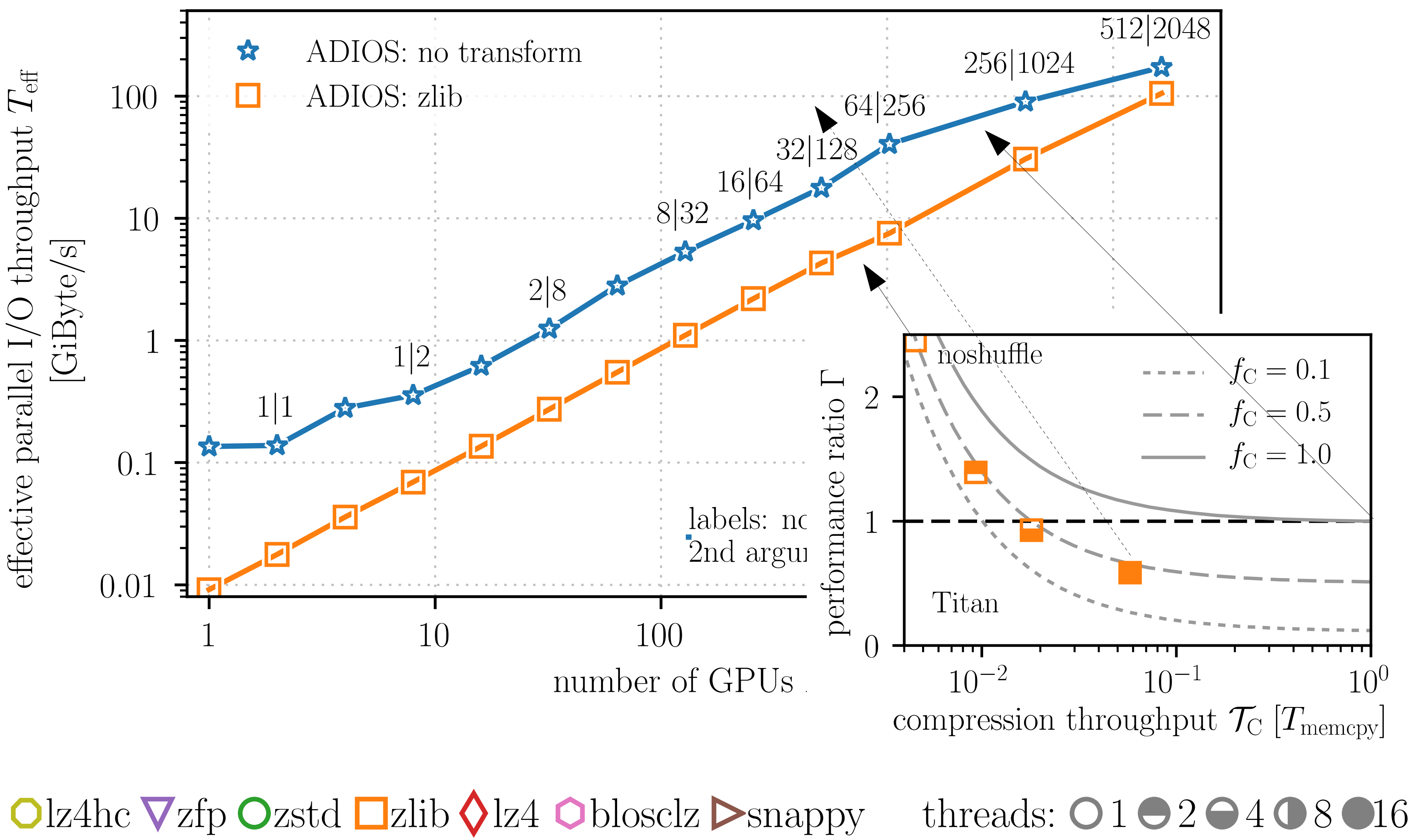
Hypnos I/O Weak Scaling

(Measured Application vs. Performance Model)



Titan I/O Weak Scaling

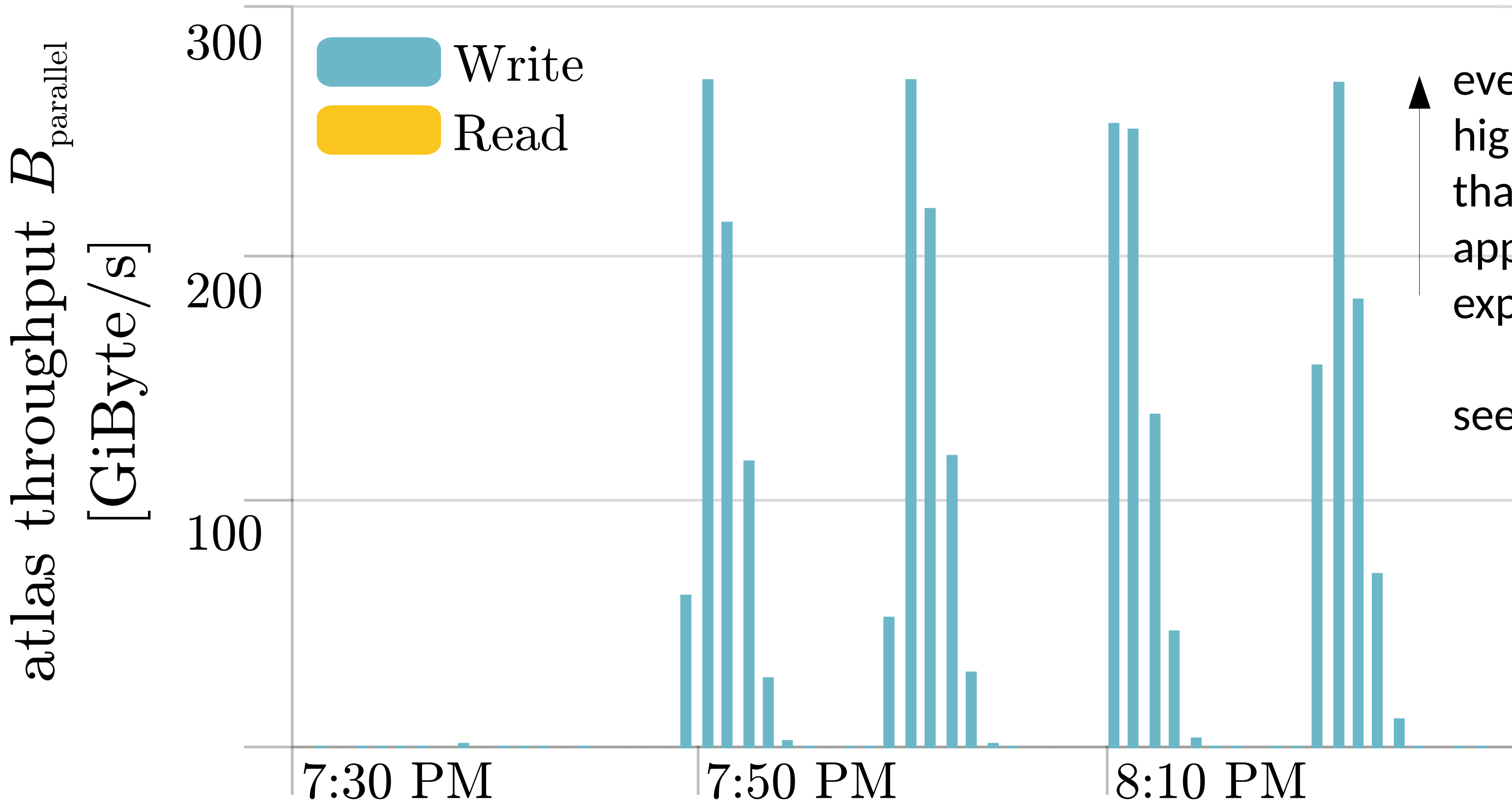
(Measured Application vs. Performance Model)



Titan I/O Weak Scaling

(Atlas Monitoring)

$$T_{\text{eff}} \equiv \frac{N \times S}{t_{\text{I/O}}} < B_{\text{parallel}}$$



even higher than app experiences

see $t_{I/O}$= ...