Generalisation of Recursive Doubling for AllReduce

Martin Rüfenacht, Mark Bull, Stephen Booth

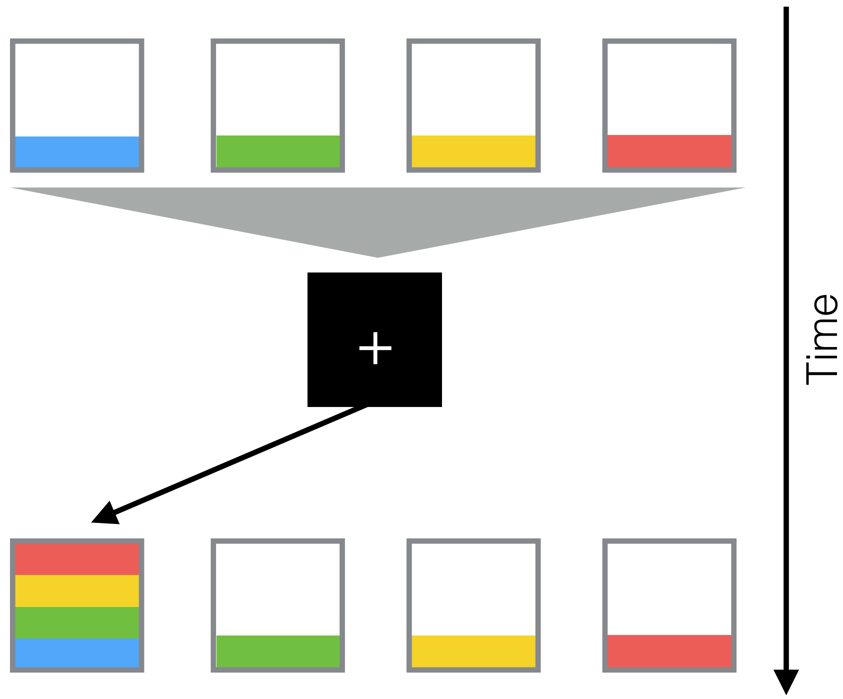
THE UNIVERSITY of EDINBURGH
EPSRC Engineering and Physical Sciences Research Council
epcc
EPSRC Centre for Doctoral Training in Pervasive Parallelism
informatics

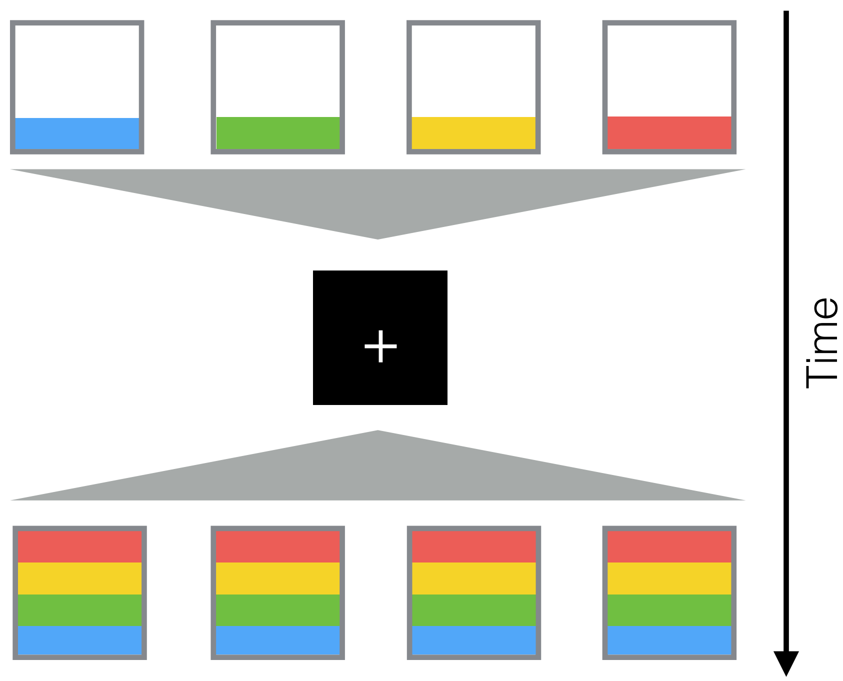
Setting the stage

- AllReduce is one of the most important MPI collectives
- AllReduce is a core dependency of iterative solvers

Operation

- **Reduction** with an operator
  - max, min, +, *, etc…
- Over networked **distributed memory** nodes
  - [2, inf)
- **All** processes need the same answer

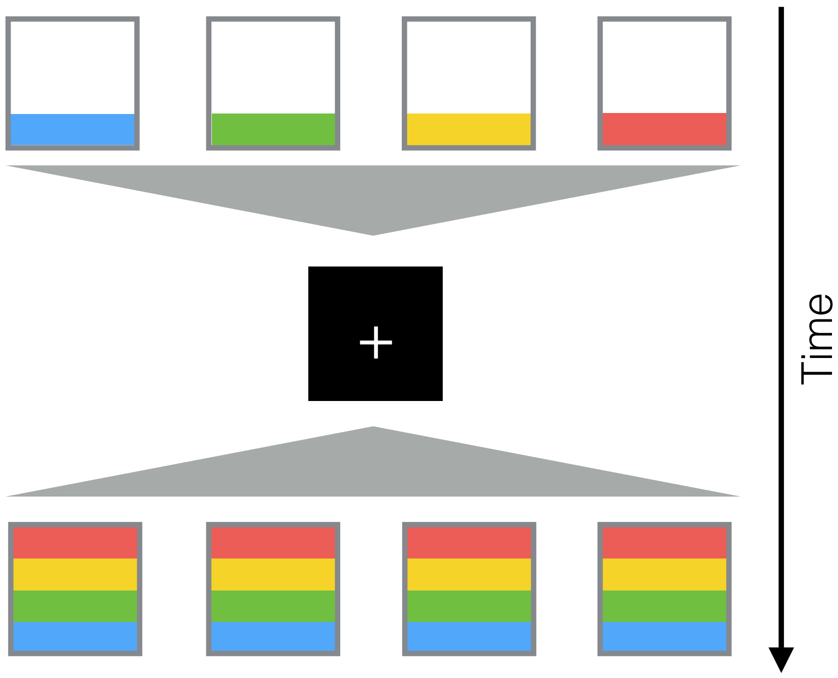


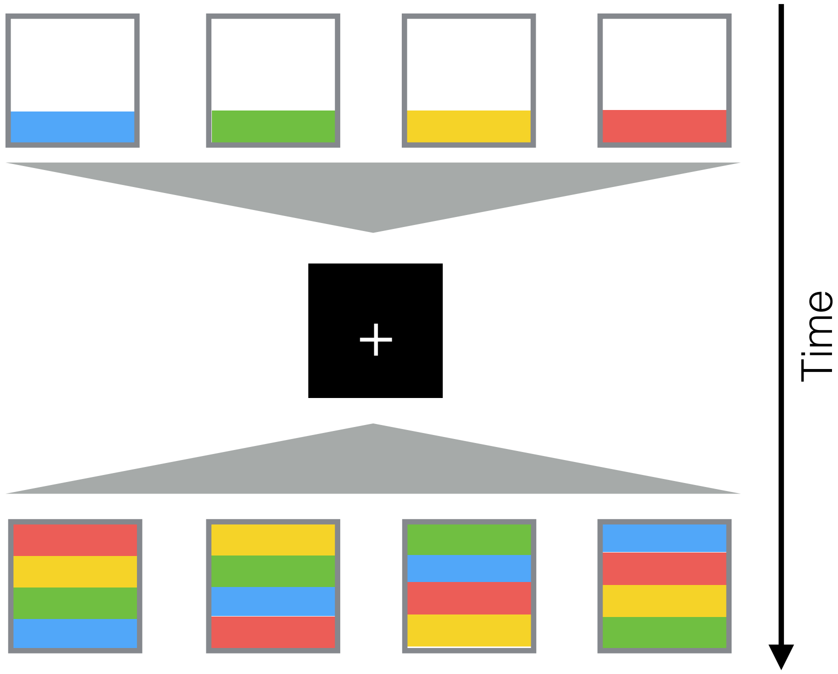


Constraints

- Consistent results required **across executions**
- Consistent results **across processes**
- **Ordering** of non-associative operators

$$(a+b)+c \neq a+(b+c)$$

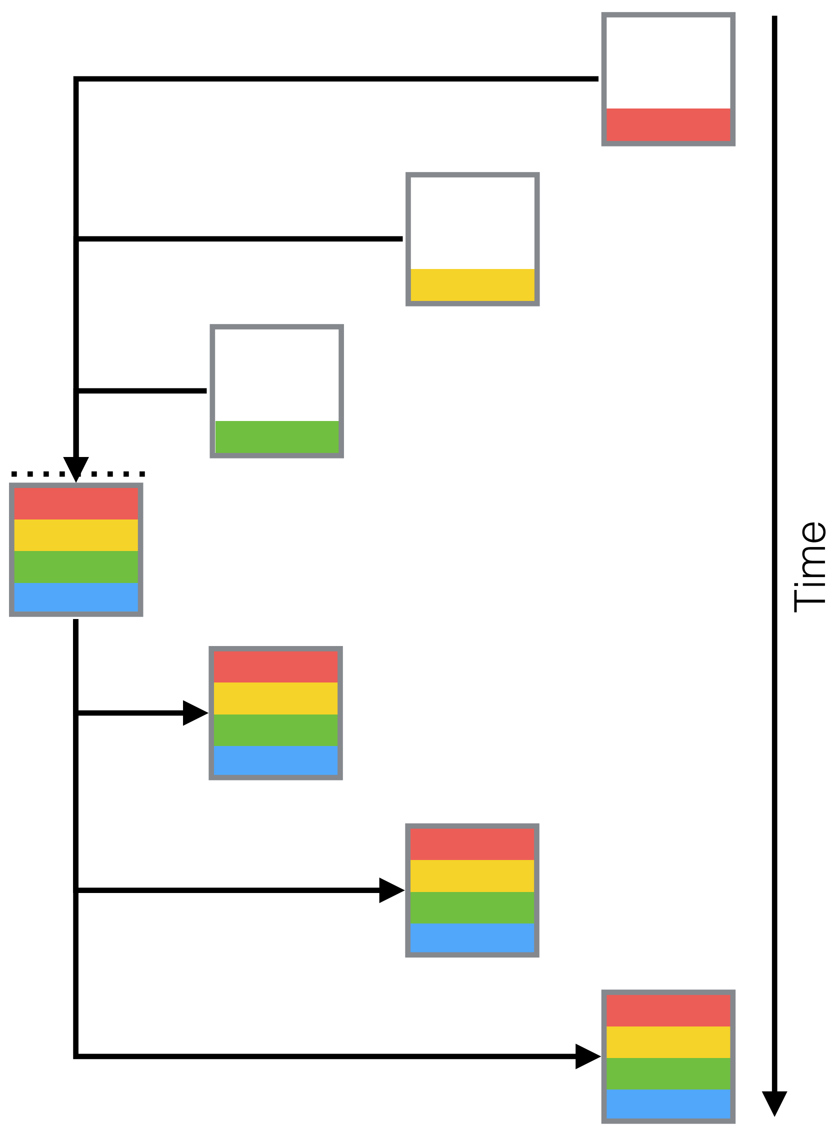




Simple Ways

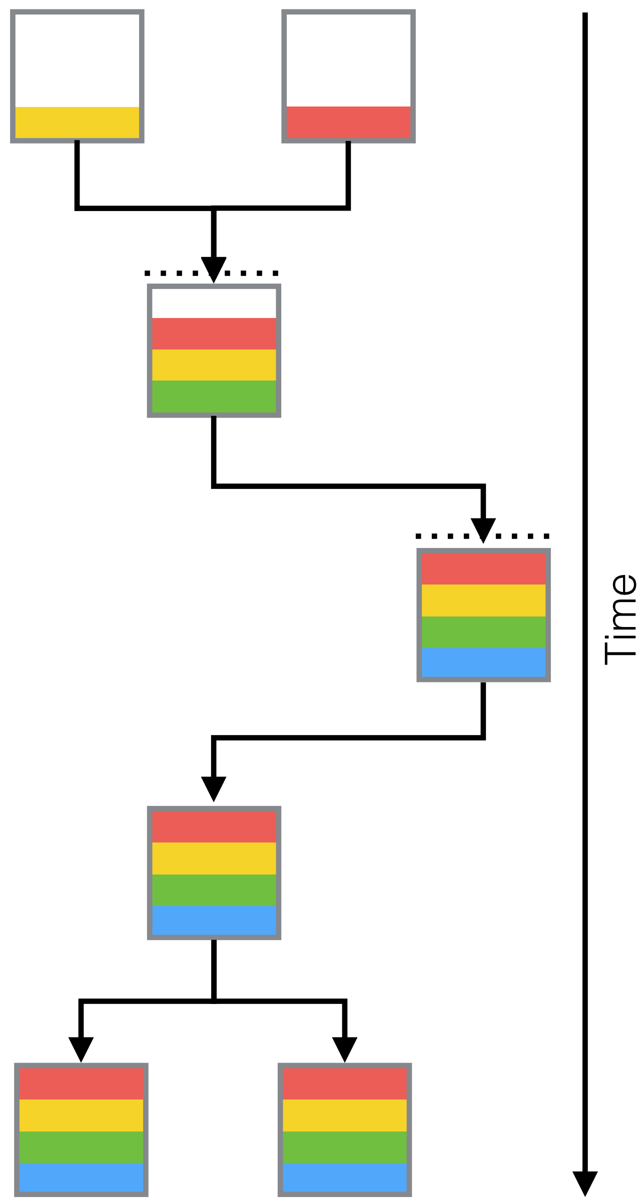
Linear Reduce & Broadcast

Time: $O(N)$



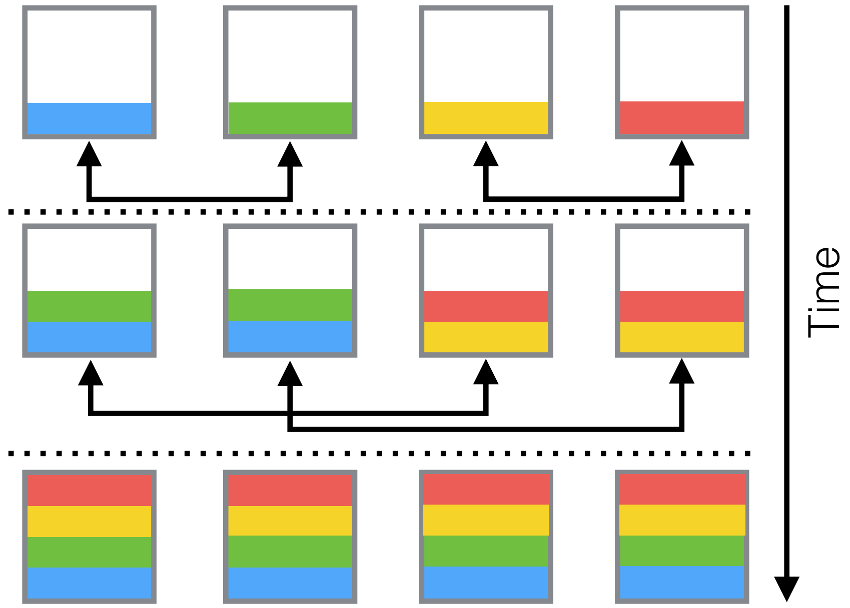
Tree Reduce & Broadcast

Time: $2\times O(\log_2 N)$

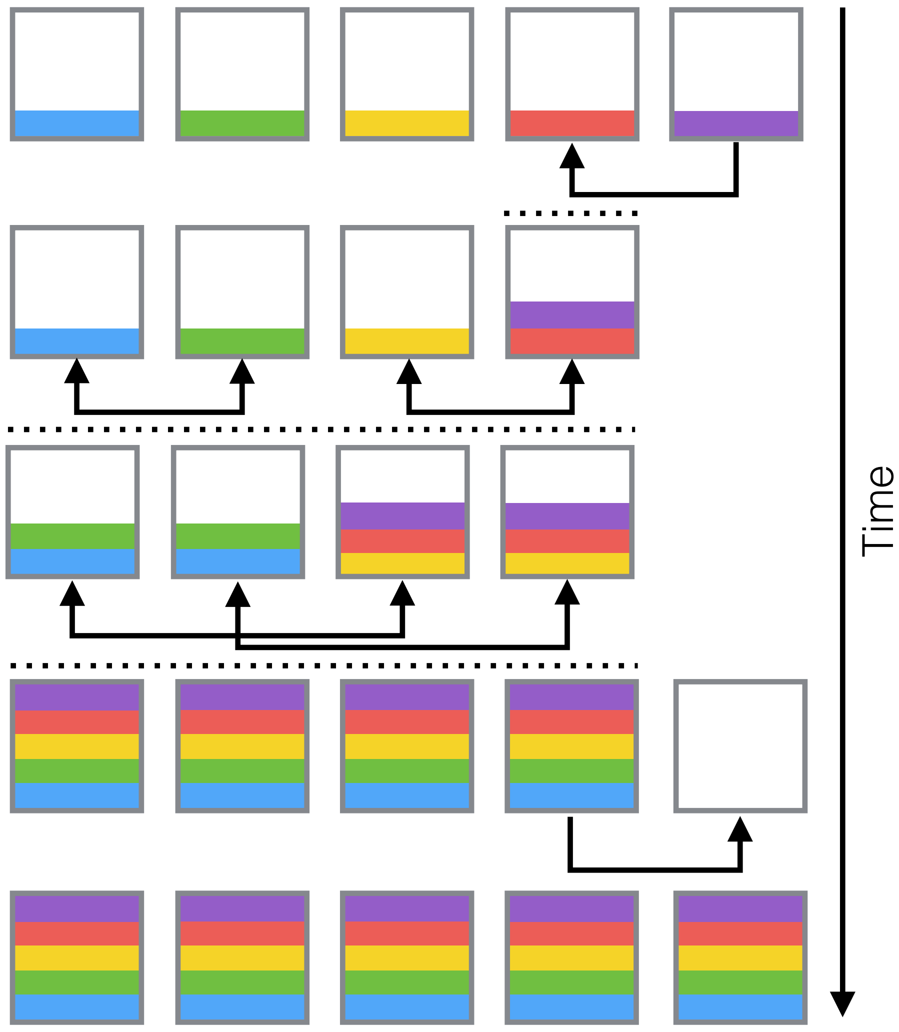


Butterfly Pattern

- Pairwise Exchange using Recursive Doubling
  - Time: $O(\log_2 N)$, requires $N = 2^k$
  - There is a fix, but costs 2 stages!

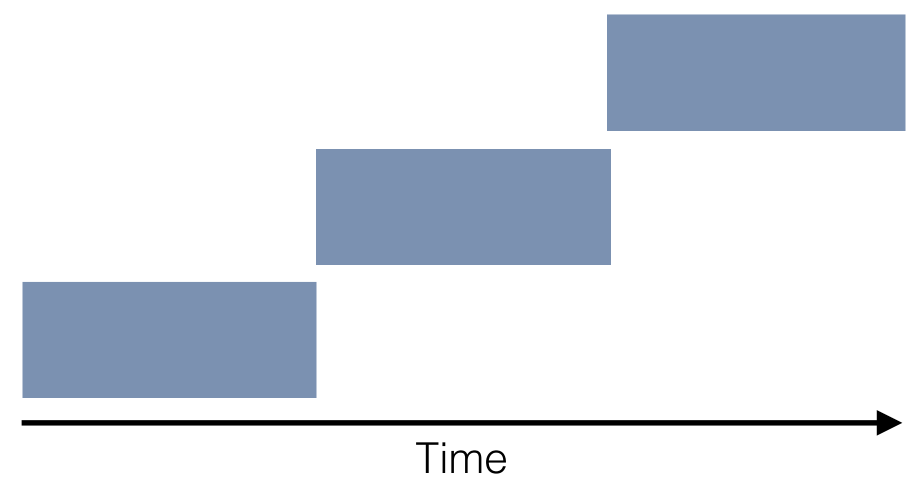


$$stages = \log_2 4 = 2$$



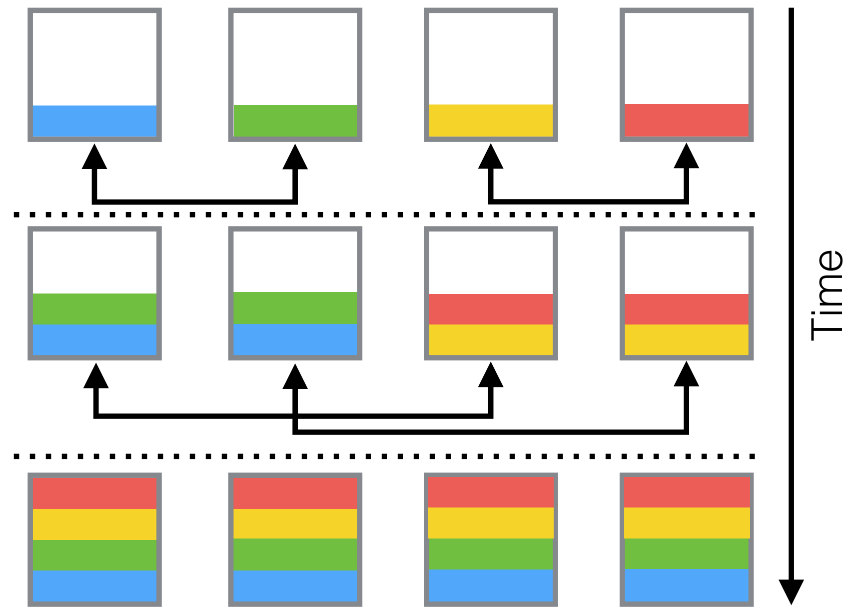
Modelling

- Simple cost model
  - For **small messages**
  - **Latency bound** messages

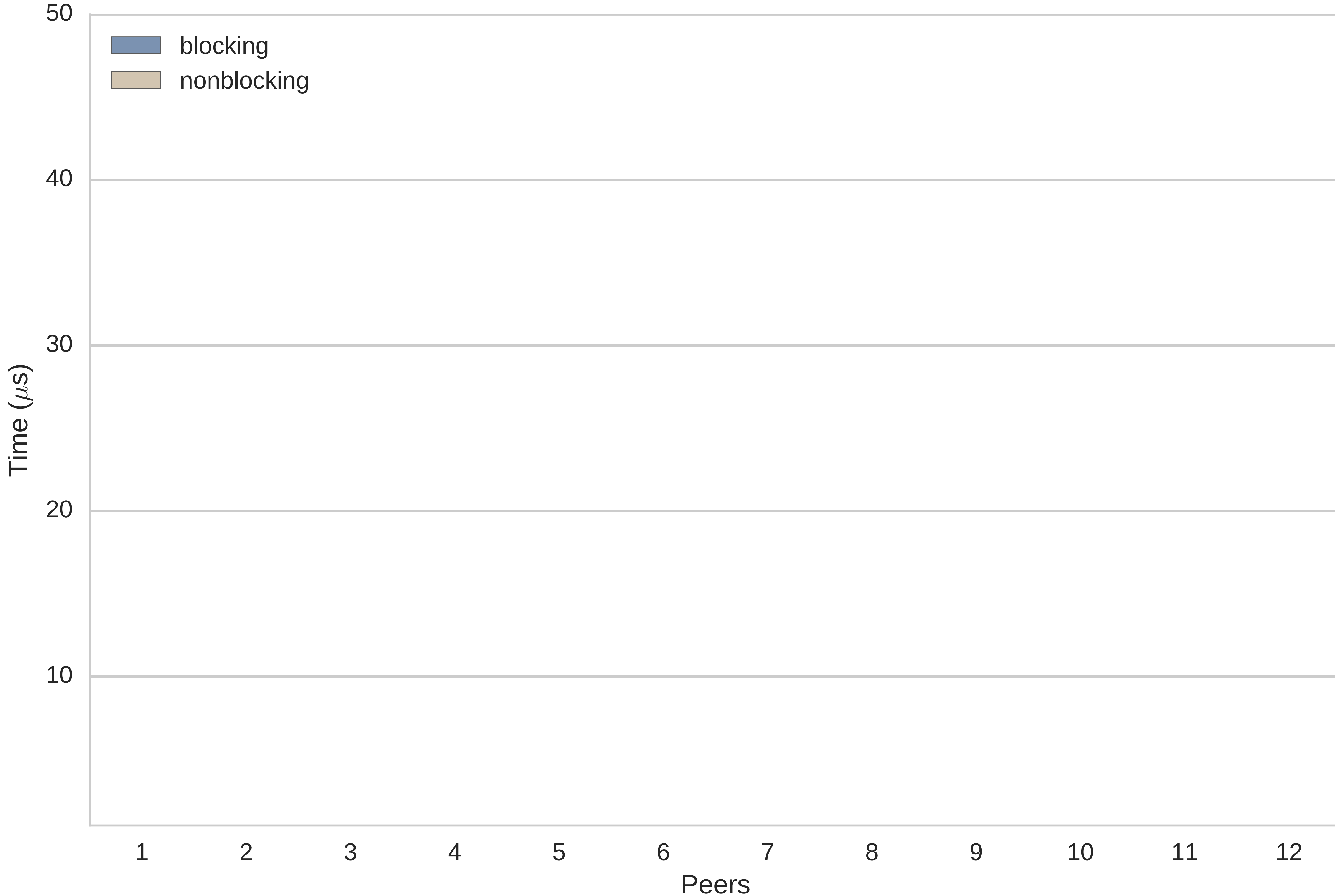


$$\alpha + \cancel{\beta n + \gamma n} \qquad n = 0$$

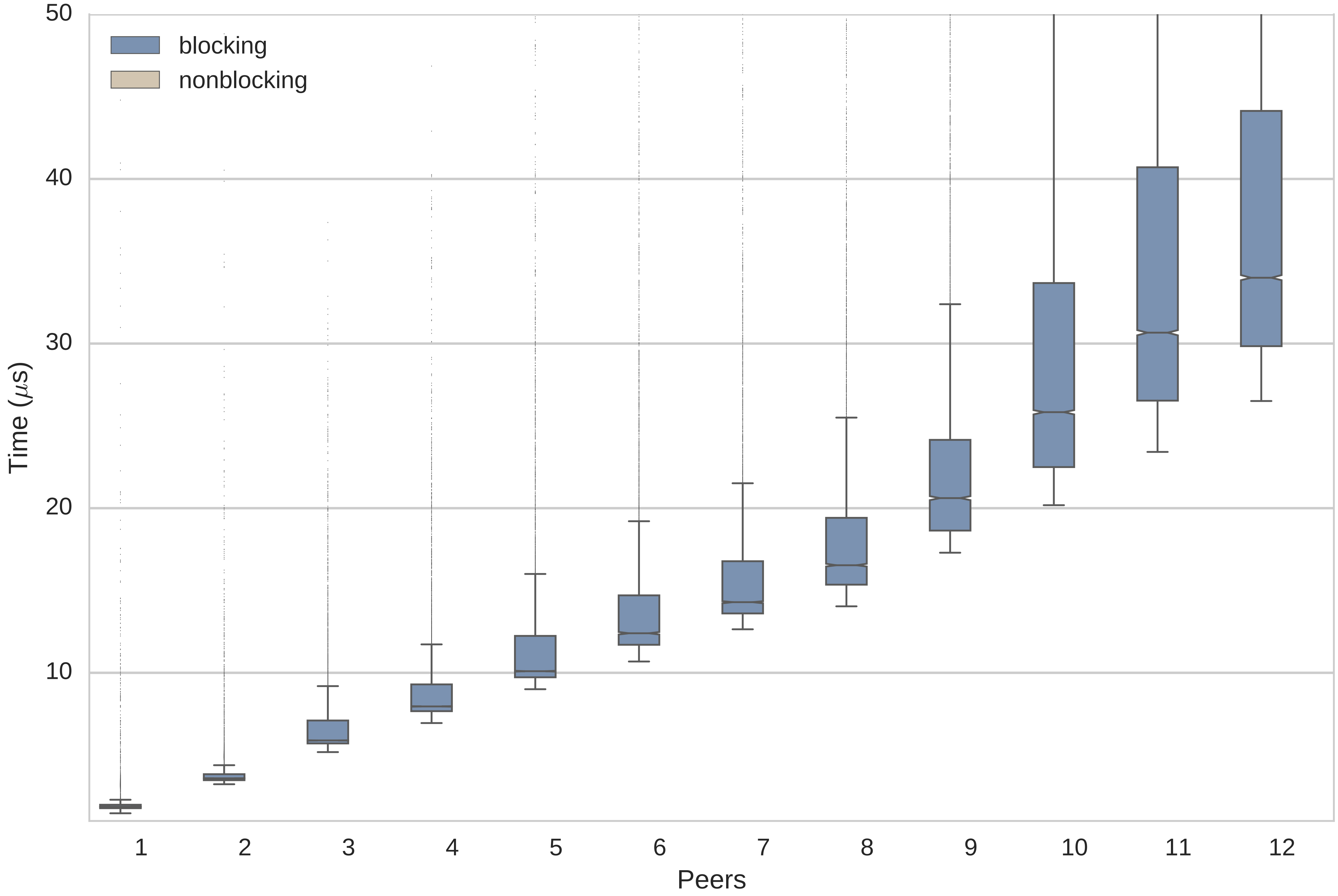
$$O(\log_2 N) \rightarrow \alpha \log_2 N$$



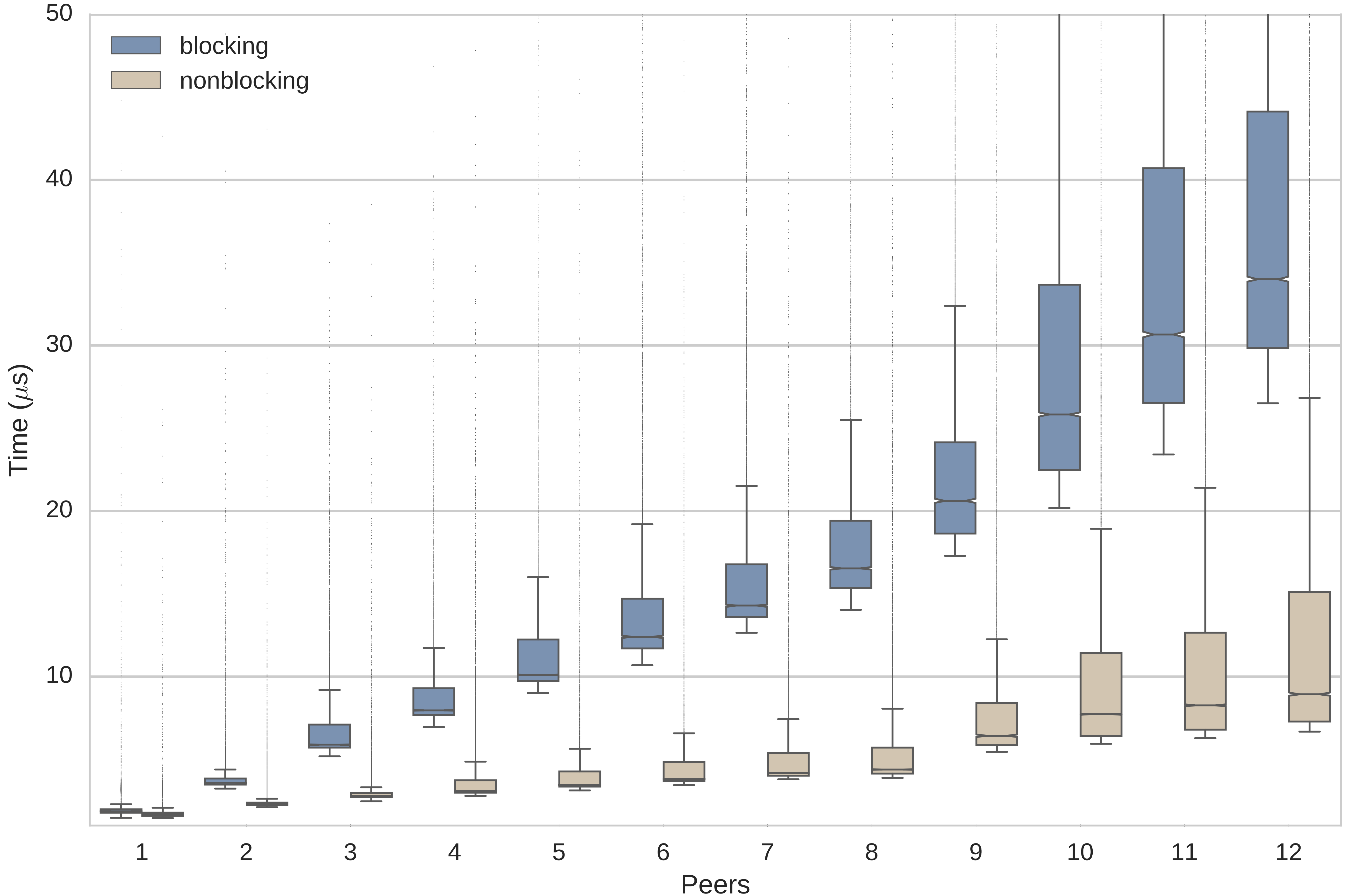
DMAPP Multicast Sends



DMAPP Multicast Sends

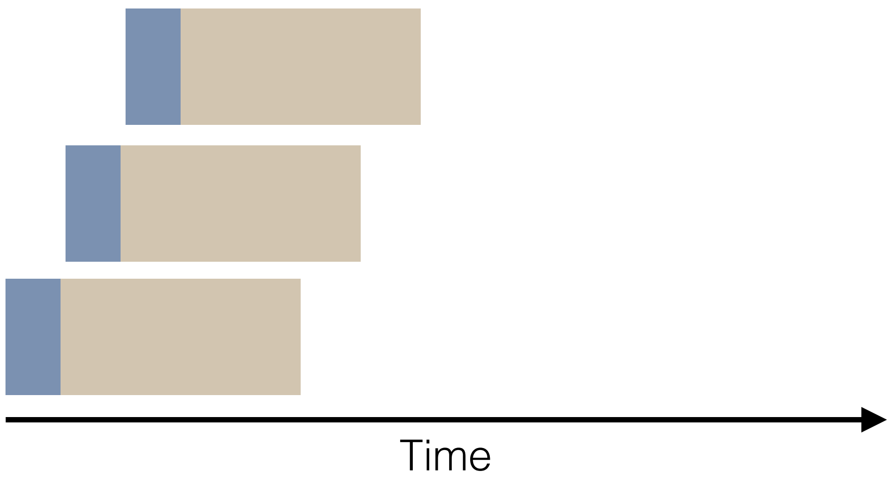


DMAPP Multicast Sends



Modelling

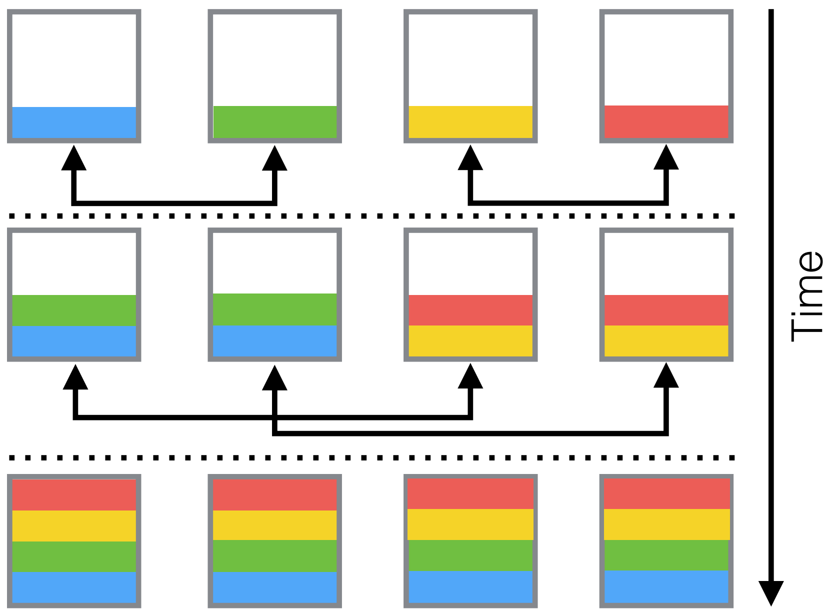
- Pipelining cost model
  - Host only sees partial cost
  - Offloaded to network
  - Results in **cheap multicasting**



$$\alpha = \alpha_p + \alpha_r$$

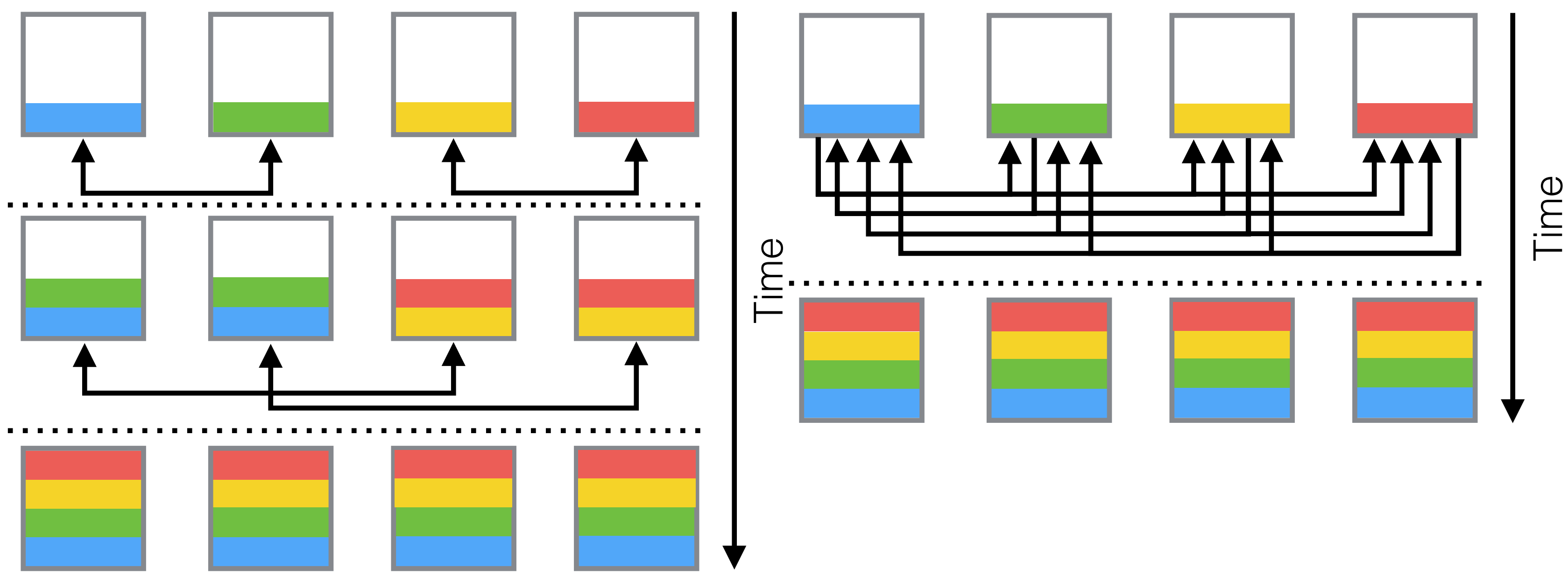
$$\alpha_p + b(\alpha_r + \cancel{\beta n + \gamma n}) \quad n = 0$$

$$(\alpha_p + b\alpha_r)\boxed{\log_2} N \rightarrow b = 1$$



First Idea

Pairwise Exchange is an **All-to-All**



Applied recursively like recursive doubling

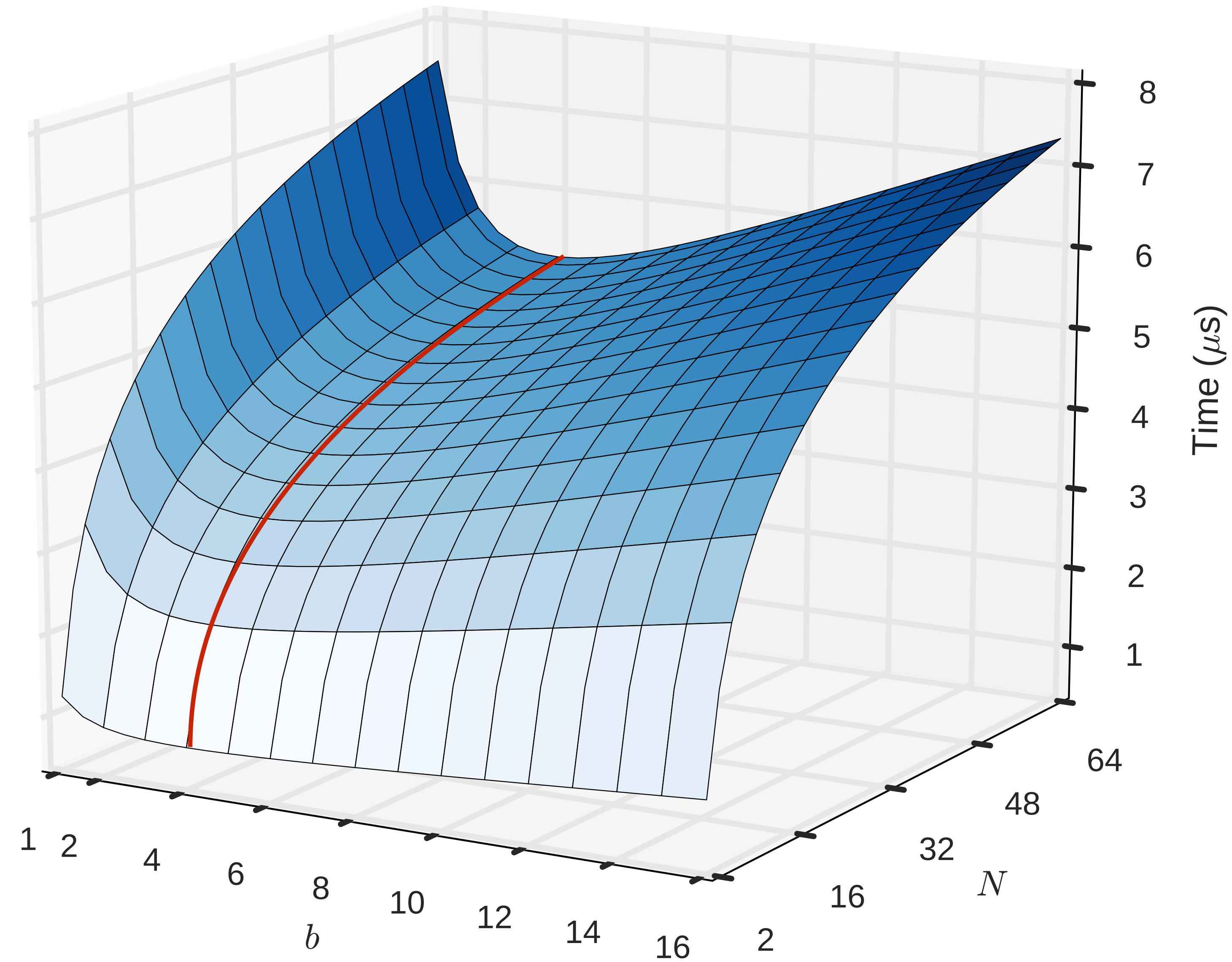
$$(\alpha_p + b\alpha_r)\log_2 N \quad \rightarrow \quad (\alpha_p + b\alpha_r)\log_{b+1} N$$

Recursive Multiplying Model

THE UNIVERSITY *of* EDINBURGH

$$(\alpha_p + b\alpha_r)\log_{b+1} N$$

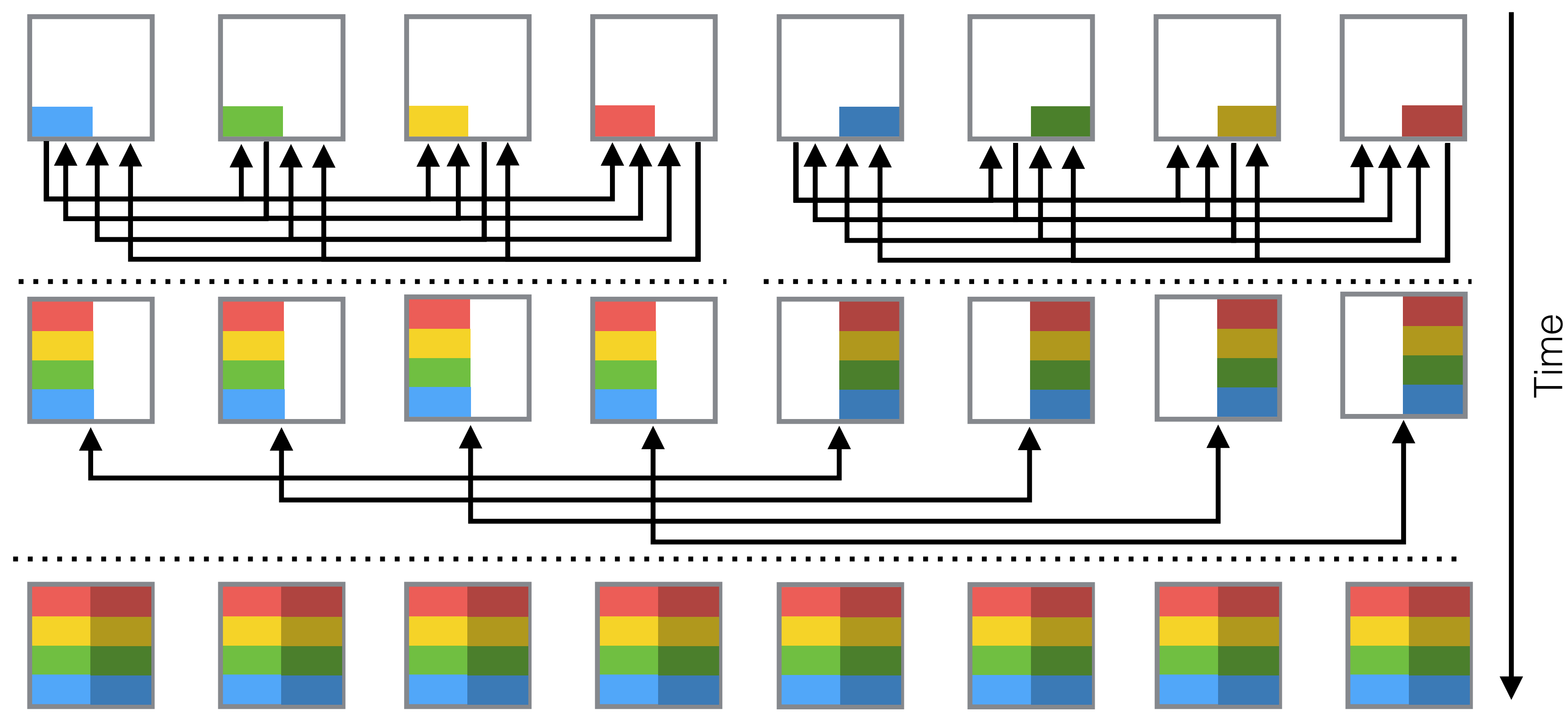
$$b_{opt} = e^{W\left(\frac{\alpha_p - \alpha_r}{e\alpha_r}\right)+1} - 1$$



N limited to powers of $b+1$

Second Idea

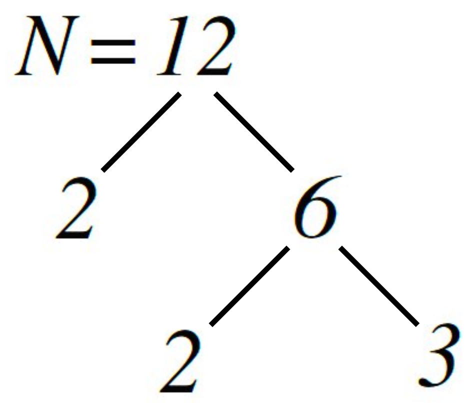
All-to-All **size can change**

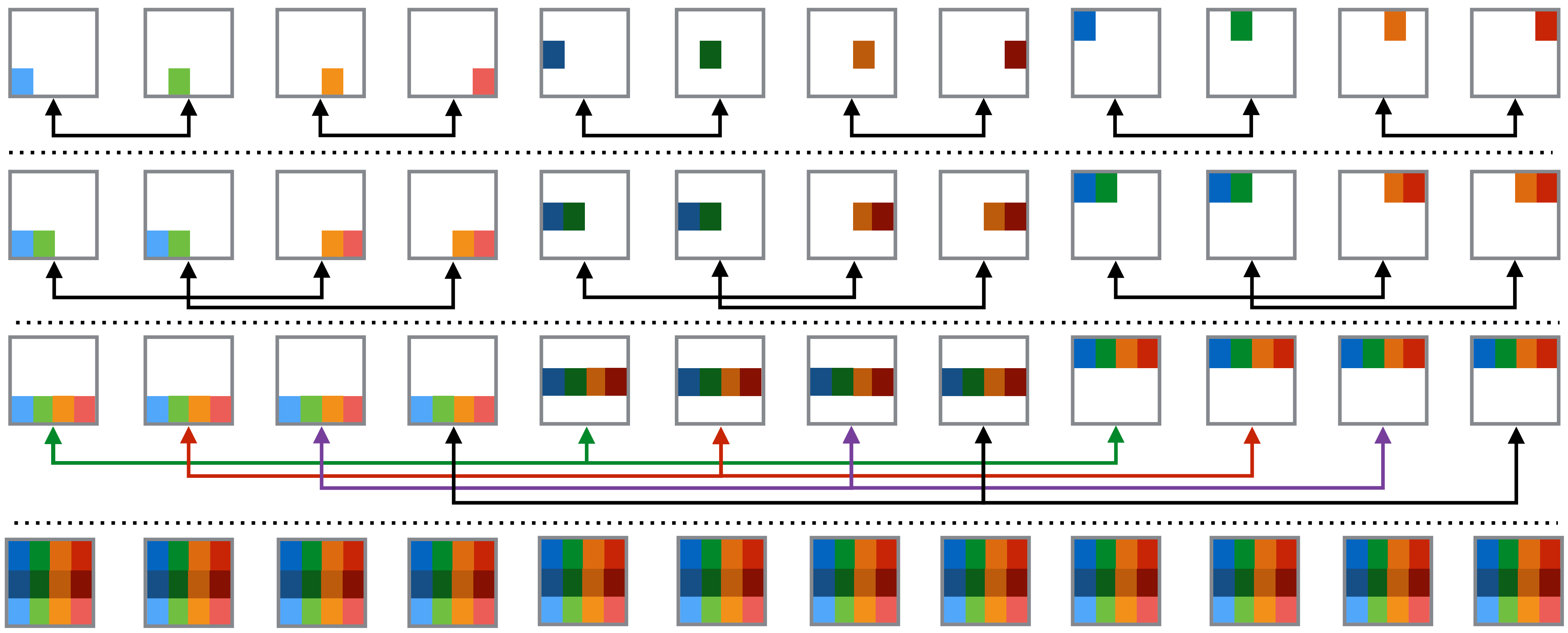


We have access to composite numbers instead of powers only

Recursive Multiplying

1. Prime factorisation of collective size

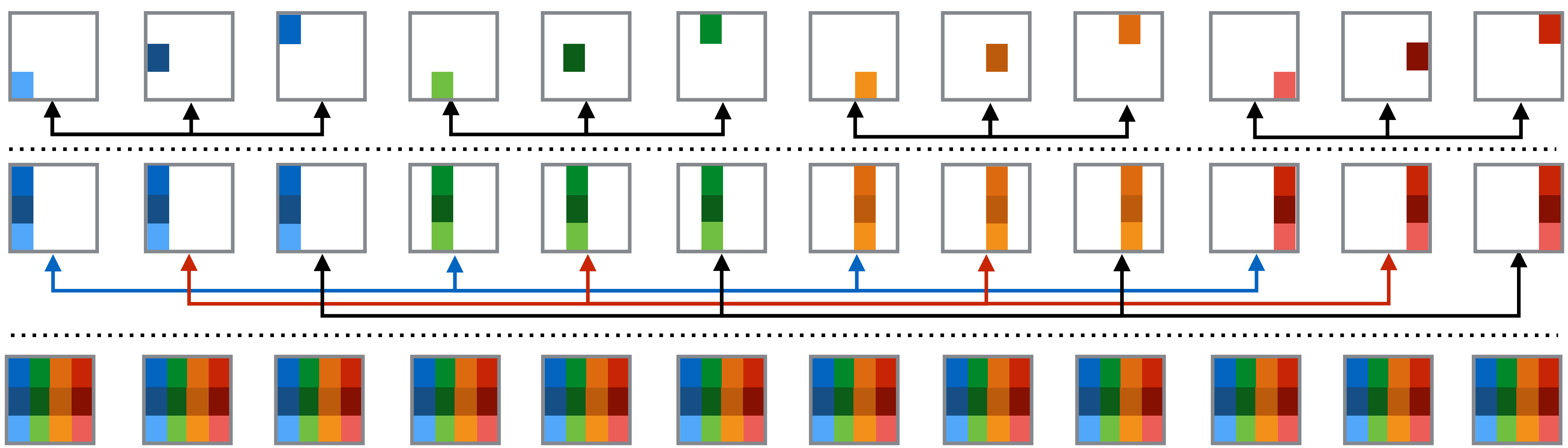




Recursive Multiplying

2. Aggregate factors
    - **Optimal** multicast usage
    - Dependent only on **overlap ratio**

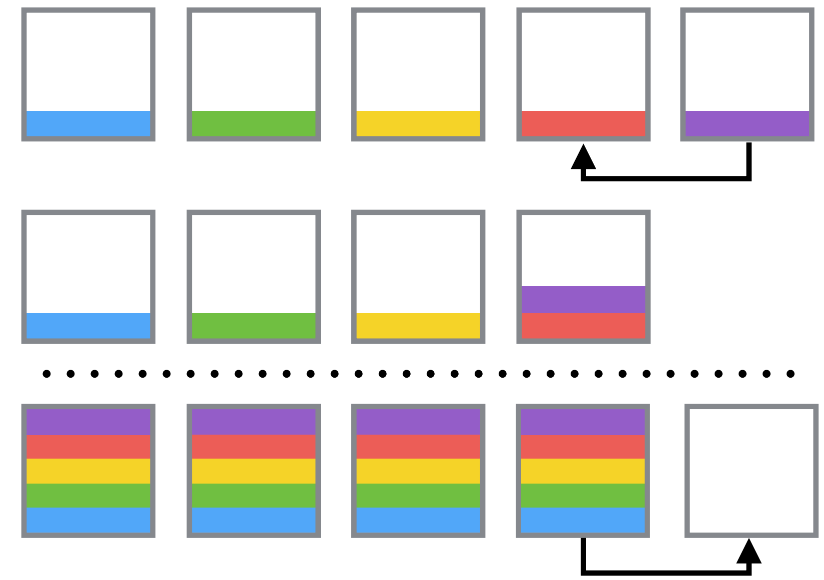
$$\left\{ \begin{array}{l} 12 \\ \boxed{3,4} \\ 2,6 \\ 2,2,3 \end{array} \right.$$

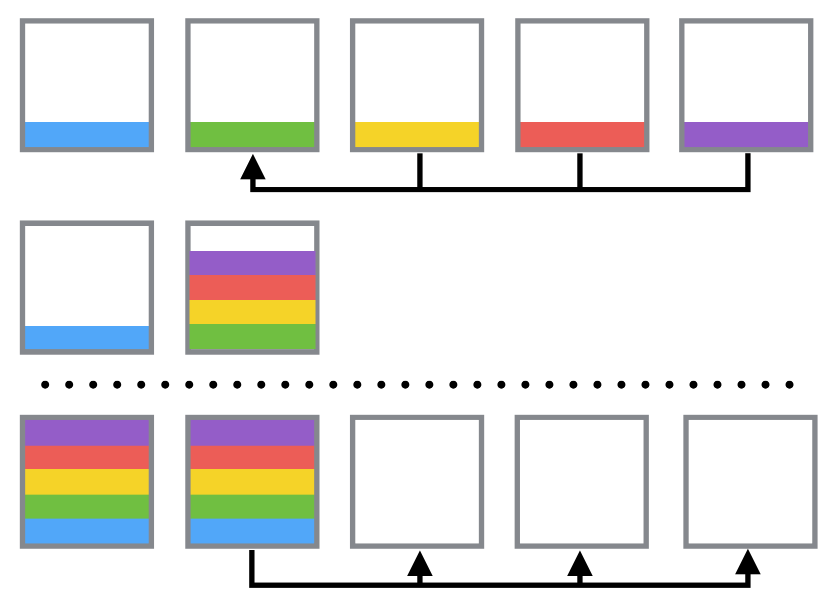


- power of 2 restriction -> large prime restriction

Splitting

- MPICH fix for the non-power-of-two case
- Make use of multicasting
- Generalised version of MPICH fix
- Still requires two additional stages



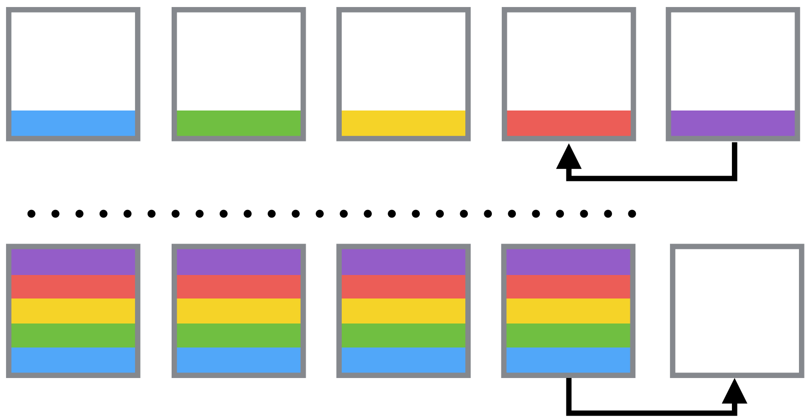


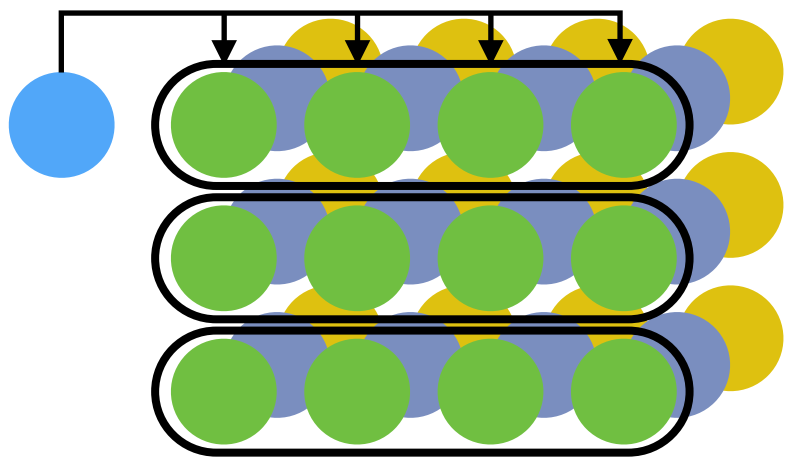
$\alpha_p + \alpha_r$

$\alpha_p + 3\alpha_r$

Merging

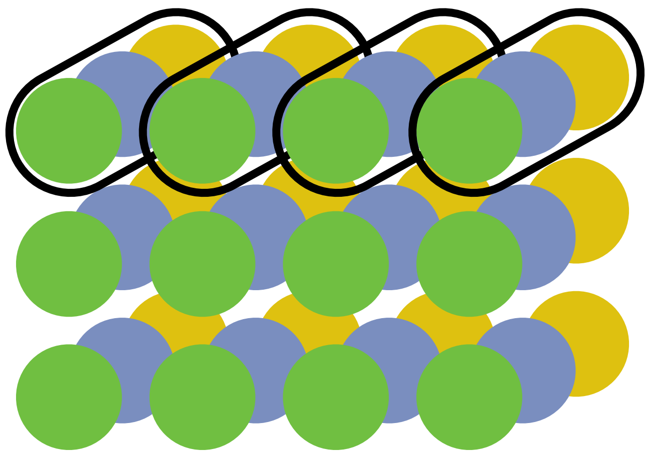
- Overlapping multiple patterns



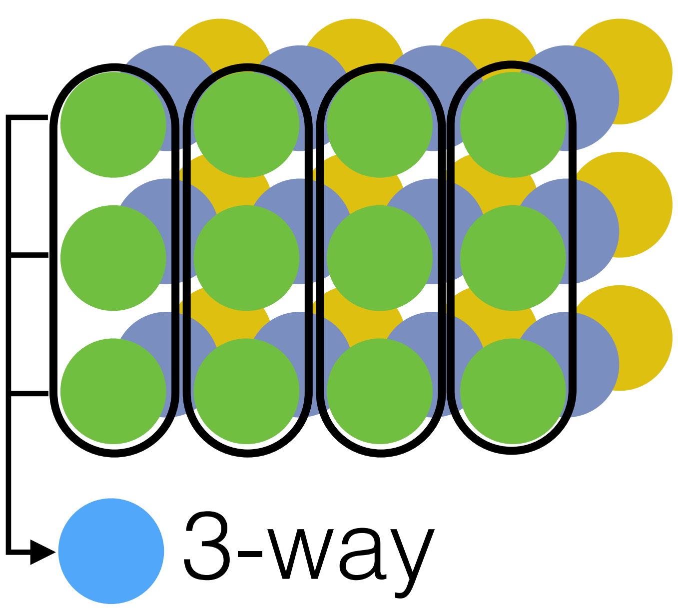


4-way

$$max(\alpha_p+4\alpha_r,\ \alpha_p+3\alpha_r)$$



3-way

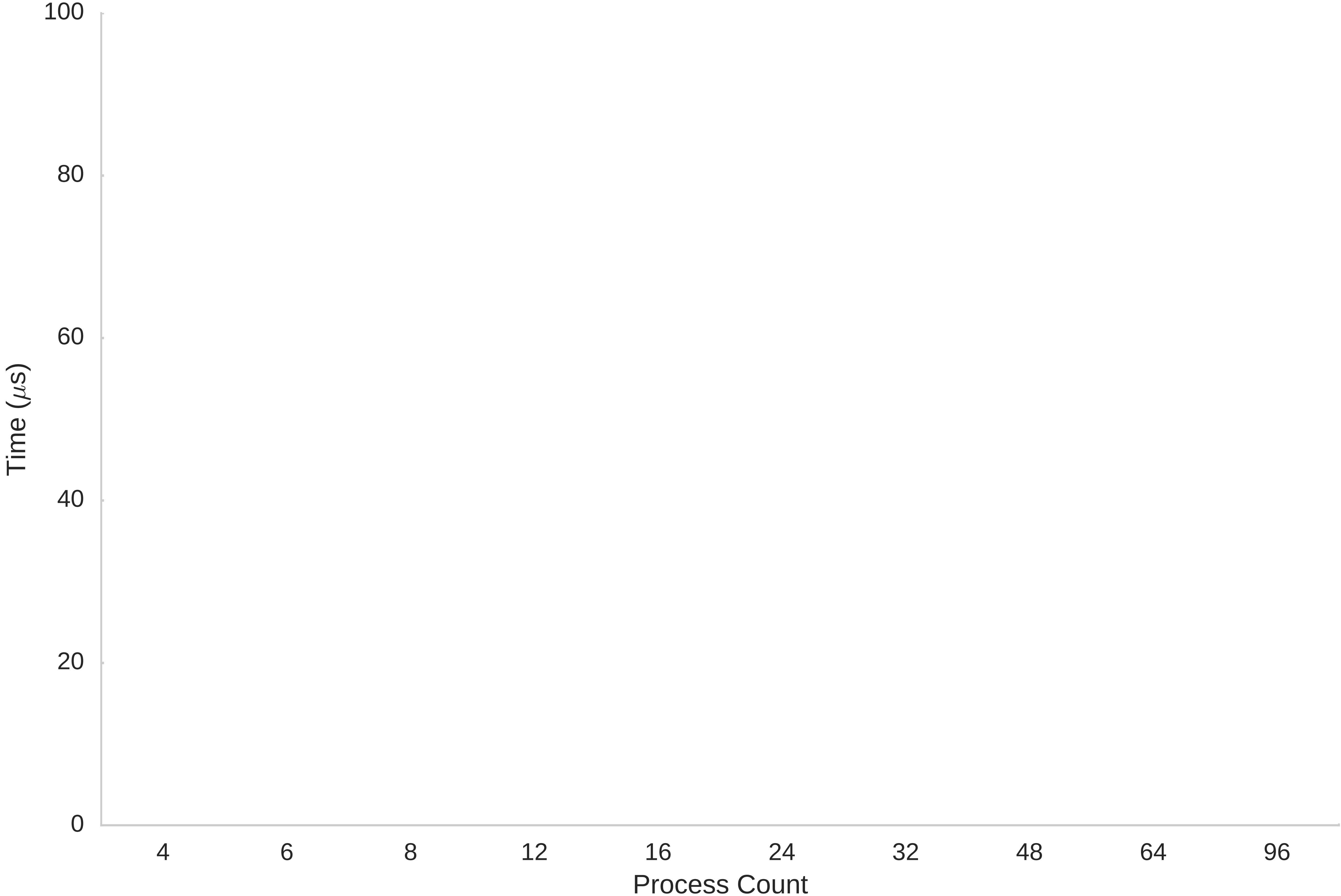


3-way

$$\alpha_p+3\alpha_r$$

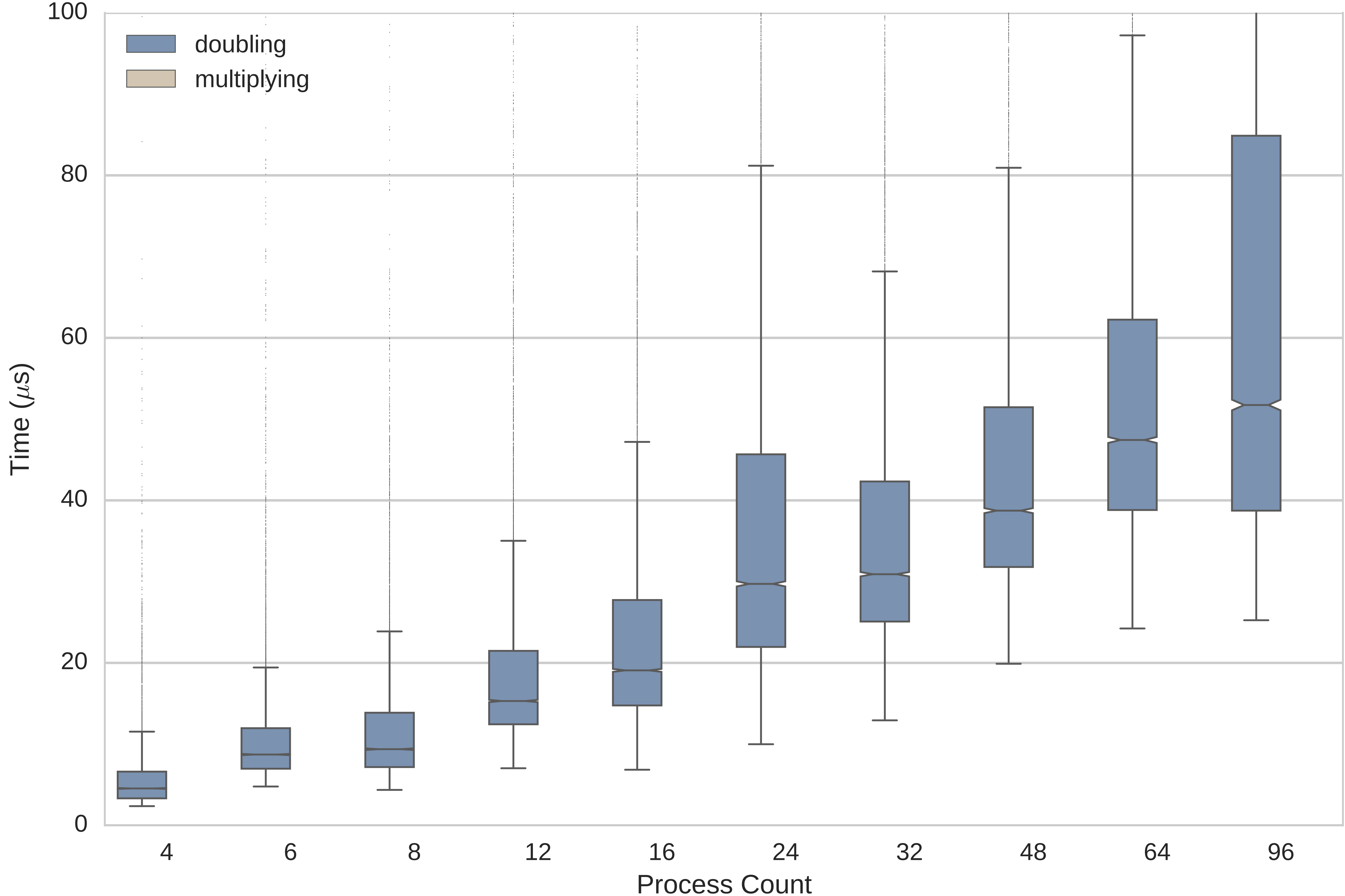
Results

THE UNIVERSITY *of* EDINBURGH



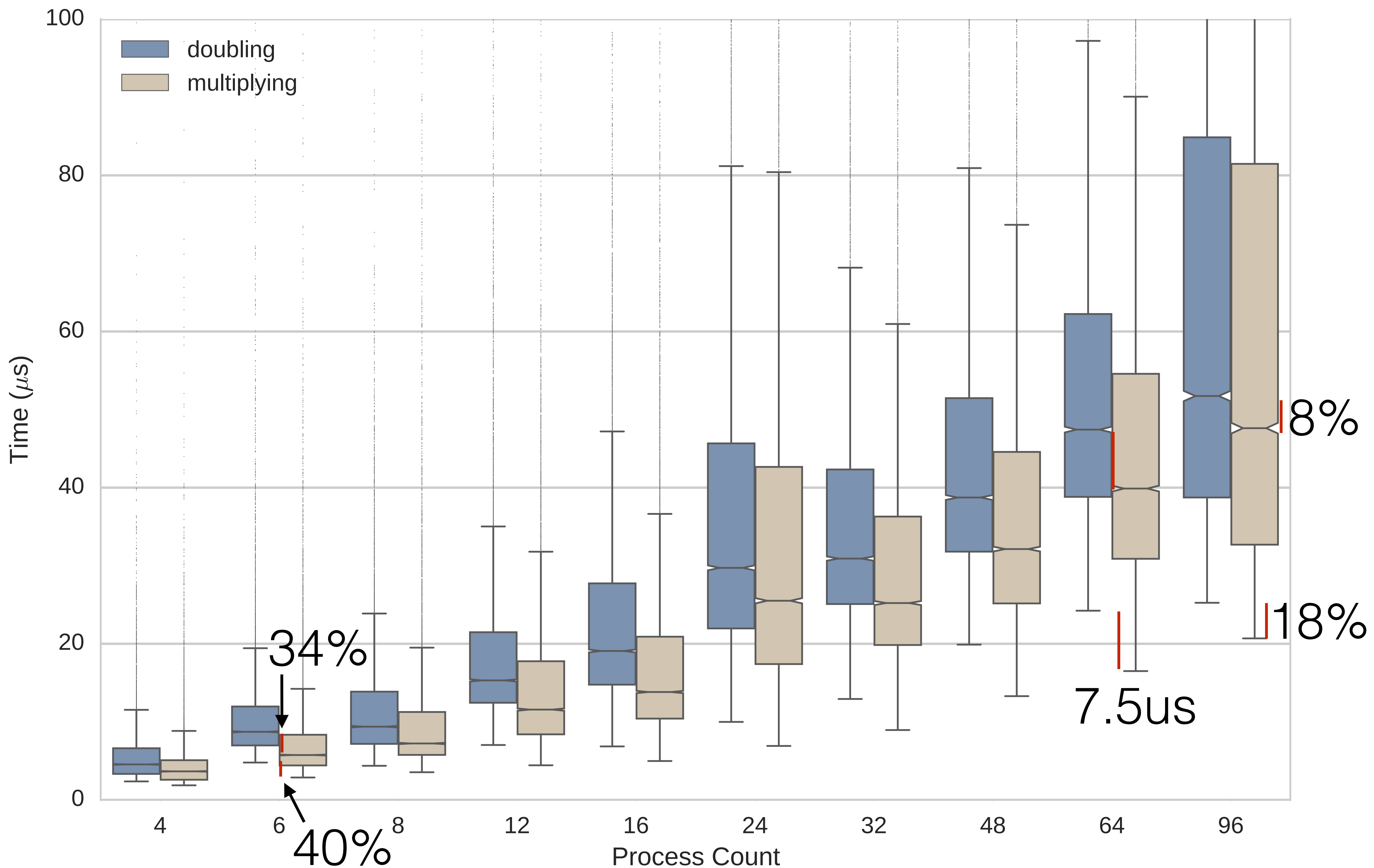
Results

THE UNIVERSITY of EDINBURGH



Results

THE UNIVERSITY of EDINBURGH



Summary

- Execution time less than recursive doubling consistently
- Drop in replacement for small message recursive doubling
- More pipelining and bandwidth