Fast Distributed Resampling and Putting More Emphasis on Modelling in Multi-Target Tracking

(Simon Maskell and) Alessandro Varsi (and Paul Spirakis)
University of Liverpool

A Varsi, S Maskell and P Spirakis.
An O(log_2N) Fully-Balanced Resampling Algorithm for Particle Filters on Distributed Memory Architectures.
Algorithms. 2021. 14(12) 342

UNIVERSITY OF LIVERPOOL

Centre for Doctoral Training in DISTRIBUTED ALGORITHMS

Motivation

Existing multi-target tracking development:

- Developing novel algorithms
- Incremental changes to the models used for the targets

Strategy being adopted in other Bayesian inference contexts:

- Using common algorithms across many (non-linear and non-Gaussian) models
- Via the widespread adoption of probabilistic programming languages (eg Stan)
- Model developments are more frequent and significant
- Improvements in performance are more frequent and significant

A note on Data Fusion architectures

- Decentralised (eg Track Fusion) is good for scalability
- Distributed (ie Autonomy) removes vulnerability to attack

UNIVERSITY OF LIVERPOOL

Centre for Doctoral Training in DISTRIBUTED ALGORITHMS

Motivation

University of Liverpool is developing Streaming-Stan

- Extension of Stan
- Streaming-Stan can cater for sequential Bayesian inference problems, eg for:
    - Multi-target tracking with joint models of all targets (integrating over data association[1])
    - Models for intent (eg future destination that is also consistent with past route)[2]
    - Integrated Expected Likelihood Particle Filter (to emulate Track-Before-Detect)[3]
    - SDEs for single objects (eg for space situational awareness/astrodynamics)
    - SDEs for pandemics

As models get more complex, real-time performance becomes more challenging

- Need to employ modern compute hardware to facilitate real-time performance

Streaming-Stan needs an efficient general-purpose algorithm

- Resampling for shared memory system (eg GPU) [4]
- First exact O (log N) resampling algorithm for distributed memory systems (e.g. multiple GPUs) [5]

[1] P Horridge and S Maskell. *Real-Time Tracking Of Hundreds Of Targets With Efficient Exact JPDAF Implementation*. Proceedings of Fusion 2006.
[2] L Vladimirov and S Maskell. *A SMC Sampler for Joint Tracking and Destination Estimation from Noisy Data*. Proc Fusion 2020
[3] A Varsi, J Taylor, L Kekempanos, E Pyzer-Knapp and S Maskell. *A Fast Parallel Particle Filter for Shared Memory Systems*. IEEE Signal Processing Letters. 2020
[4] M Ransom, L Vladimirov, P Horridge, J F Ralph and S Maskell. Integrated Expected Likelihood Particle Filters. Proc Fusion 2020
[5] A Varsi, S Maskell and P Spirakis. *An $O(\log_2 N)$ Fully-Balanced Resampling Algorithm for Particle Filters on Distributed Memory Architectures*. Algorithms. 2021. 14(12) 342

UNIVERSITY OF LIVERPOOL

Centre for Doctoral Training in DISTRIBUTED ALGORITHMS

Context

Focus is on resampling

- Algorithmic component at the heart of particle filters (and SMC samplers)

Focus in on a distributed memory setting

- Have to think in terms of message-passing
- $O(\log_2 N)$ demands a divide-and-conquer approach
    - Best pre-existing approach is $O((\log_2 N)^2)$
- Some components (eg element-wise operations, sum and cumulative-sum) “easy”

Let’s consider an example with 16 particles …

Overarching Strategy

Consider some data:

Data	AA	BB	CC	DD	EE	FF	GG	HH	II	JJ	KK	LL	MM	NN	OO	PP
Ncopies	5	0	1	0	0	5	0	0	0	0	4	0	1	0	0	0

Step 1: Push all the data with non-zero Ncopies to the left:

Data	AA	CC	FF	KK	MM	PP	X	X	X	X	X	X	X	X	X	X
Ncopies	5	1	5	4	1	0	0	0	0	0	0	0	0	0	0	0

Step 2: Push the data to the right to make gaps:

Data	AA	X	X	X	X	CC	FF	X	X	X	X	KK	X	X	X	MM
Ncopies	5	0	0	0	0	1	5	0	0	0	0	4	0	0	0	1

Step 3: Populate the gaps:

Data	AA	AA	AA	AA	AA	CC	FF	FF	FF	FF	FF	KK	KK	KK	KK	MM
Ncopies	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

UNIVERSITY OF LIVERPOOL

Centre for Doctoral Training in DISTRIBUTED ALGORITHMS

Step 1: Push the non-zeros to the left (1/2)

Consider some data:

Data	AA	BB	CC	DD	EE	FF	GG	HH	II	JJ	KK	LL	MM	NN	OO	PP
Ncopies	5	0	1	0	0	5	0	0	0	0	4	0	1	0	0	0

Step 1.1: Work out how many zeros to the left of each datum and represent in binary and shift by bit 1

Binary	0000	0000	0001	0001	0010	0011	0011	0100	0101	0110	0111	0111	1000	1000	1001	1010
Data	AA	CC	DD	X	FF	GG	X	II	X	KK	LL	X	MM	OO	X	PP
Ncopies	5	1	0	0	5	0	0	0	0	4	0	0	1	0	0	0

UNIVERSITY OF LIVERPOOL

Centre for Doctoral Training in DISTRIBUTED ALGORITHMS

Step 1: Push the non-zeros to the left (1/2)

Consider some data:

Data	AA	BB	CC	DD	EE	FF	GG	HH	II	JJ	KK	LL	MM	NN	OO	PP
Ncopies	5	0	1	0	0	5	0	0	0	0	4	0	1	0	0	0

Step 1.1: Work out how many zeros to the left of each datum and represent in binary and shift by bit 1

Binary	0000	0000	0001	0001	0010	0011	0011	0100	0101	0110	0111	0111	1000	1000	1001	1010
Data	AA	CC	DD	X	FF	GG	X	II	X	KK	LL	X	MM	OO	X	PP
Ncopies	5	1	0	0	5	0	0	0	0	4	0	0	1	0	0	0

Step 1.2: Work out how many zeros to the left of each datum and represent in binary and shift by bit 2:

Binary	0000	0000	0000	0001	0010	0010	0011	0100	0101	0110	0110	0111	1000	1000	1001	1010
Data	AA	CC	FF	GG	X	X	X	KK	LL	X	X	X	MM	PP	X	X
Ncopies	5	1	5	0	0	0	0	4	0	0	0	0	1	0	0	0

Step 1: Push the non-zeros to the left (2/2)

Output from Step 1.2:

Data	AA	CC	FF	GG	X	X	X	KK	LL	X	X	X	MM	PP	X	X
Ncopies	5	1	5	0	0	0	0	4	0	0	0	0	1	0	0	0

Step 1.3: Work out how many zeros to the left of each datum and represent in binary and shift by bit 3

Binary	0000	0000	0000	0000	0001	0010	0011	0100	0100	0101	0110	0111	1000	1000	1001	1010
Data	AA	CC	FF	KK	LL	X	X	X	X	X	X	X	MM	PP	X	X
Ncopies	5	1	5	4	0	0	0	0	0	0	0	0	1	0	0	0

UNIVERSITY OF LIVERPOOL

Centre for Doctoral Training in DISTRIBUTED ALGORITHMS

Step 1: Push the non-zeros to the left (2/2)

Output from Step 1.2:

Data	AA	CC	FF	GG	X	X	X	KK	LL	X	X	X	MM	PP	X	X
Ncopies	5	1	5	0	0	0	0	4	0	0	0	0	1	0	0	0

Step 1.3: Work out how many zeros to the left of each datum and represent in binary and shift by bit 3

Binary	0000	0000	0000	0000	0001	0010	0011	0100	0100	0101	0110	0111	1000	1000	1001	1010
Data	AA	CC	FF	KK	LL	X	X	X	X	X	X	X	MM	PP	X	X
Ncopies	5	1	5	4	0	0	0	0	0	0	0	0	1	0	0	0

Step 1.4: Work out how many zeros to the left of each datum and represent in binary and shift by bit 4:

Binary	0000	0000	0000	0000	0000	0001	0010	0011	0100	0101	0110	0111	1000	1000	1001	1010
Data	AA	CC	FF	KK	MM	PP	X	X	X	X	X	X	X	X	X	X
Ncopies	5	1	5	4	1	0	0	0	0	0	0	0	0	0	0	0

Step 2: Make the gaps (1/2)

Output from step 1:

Data	AA	CC	FF	KK	MM	PP	X	X	X	X	X	X	X	X	X	X
Ncopies	5	1	5	4	1	0	0	0	0	0	0	0	0	0	0	0

Step 2.1: Work out how much each datum needs to shift and represent in binary and shift by bit 4

Binary	0000	0100	0100	1000	1011	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
Data	AA	CC	FF	X	X	PP	X	X	X	X	X	KK	MM	X	X	X
Ncopies	5	1	5	0	0	0	0	0	0	0	0	4	1	0	0	0

Step 2: Make the gaps (1/2)

Output from step 1:

Data	AA	CC	FF	KK	MM	PP	X	X	X	X	X	X	X	X	X	X
Ncopies	5	1	5	4	1	0	0	0	0	0	0	0	0	0	0	0

Step 2.1: Work out how much each datum needs to shift and represent in binary and shift by bit 4

Binary	0000	0100	0100	1000	1011	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
Data	AA	CC	FF	X	X	PP	X	X	X	X	X	KK	MM	X	X	X
Ncopies	5	1	5	0	0	0	0	0	0	0	0	4	1	0	0	0

Step 2.2: Work out how much each datum needs to shift and represent in binary and shift by bit 3

Binary	0000	0100	0100	0000	0000	0000	0000	0000	0000	0000	0000	0000	0011	0000	0000	0000
Data	AA	X	X	X	X	CC	FF	X	X	X	X	KK	MM	X	X	X
Ncopies	5	0	0	0	0	1	5	0	0	0	0	4	1	0	0	0

UNIVERSITY OF LIVERPOOL

Centre for Doctoral Training in DISTRIBUTED ALGORITHMS

Step 2: Make the gaps (2/2)

Output from Step 2.2:

Data	AA	X	X	X	X	CC	FF	X	X	X	X	KK	MM	X	X	X
Ncopies	5	0	0	0	0	1	5	0	0	0	0	4	1	0	0	0

Step 2.3: Work out how much each datum needs to shift and represent in binary and shift by bit 2

Binary	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0011	0000	0000	0000
Data	AA	X	X	X	X	CC	FF	X	X	X	X	KK	X	X	MM	X
Ncopies	5	0	0	0	0	1	5	0	0	0	0	4	0	0	1	0

Step 2: Make the gaps (2/2)

Output from Step 2.2:

Data	AA	X	X	X	X	CC	FF	X	X	X	X	KK	MM	X	X	X
Ncopies	5	0	0	0	0	1	5	0	0	0	0	4	1	0	0	0

Step 2.3: Work out how much each datum needs to shift and represent in binary and shift by bit 2

Binary	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0011	0000	0000	0000
Data	AA	X	X	X	X	CC	FF	X	X	X	X	KK	X	X	MM	X
Ncopies	5	0	0	0	0	1	5	0	0	0	0	4	0	0	1	0

Step 2.3: Work out how much each datum needs to shift and represent in binary and shift by bit 1

Binary	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0001	0000
Data	AA	X	X	X	X	CC	FF	X	X	X	X	KK	X	X	X	MM
Ncopies	5	0	0	0	0	1	5	0	0	0	0	4	0	0	0	1

Step 3: Fill the gaps (1/2)

Output from step 2:

Data	AA	X	X	X	X	CC	FF	X	X	X	X	KK	X	X	X	MM
Ncopies	5	0	0	0	0	1	5	0	0	0	0	4	0	0	0	1

Step 3.1: Express Ncopies-1 in binary and split according to bit 4

Binary	0100	0000	0000	0000	0000	0000	0100	0000	0000	0000	0000	0011	0000	0000	0000	0000
Data	AA	X	X	X	X	CC	FF	X	X	X	X	KK	X	X	X	MM
Ncopies	5	0	0	0	0	1	5	0	0	0	0	4	0	0	0	1

Step 3.2: Express Ncopies-1 in binary and split according to bit 3

Binary	0100	0000	0000	0000	0000	0000	0100	0000	0000	0000	0000	0011	0000	0000	0000	0000
Data	AA	X	X	X	AA	CC	FF	X	X	X	FF	KK	X	X	X	MM
Ncopies	4	0	0	0	1	1	4	0	0	0	1	4	0	0	0	1

UNIVERSITY OF LIVERPOOL

Centre for Doctoral Training in DISTRIBUTED ALGORITHMS

Step 3: Fill the gaps (2/2)

Output from Step 3.2:

Data	AA	X	X	X	AA	CC	FF	X	X	X	FF	KK	X	X	X	MM
Ncopies	4	0	0	0	1	1	4	0	0	0	1	4	0	0	0	1

Step 3.3: Express Ncopies-1 in binary and split according to bit 2

Binary	0011	0000	0000	0000	0000	0000	0011	0000	0000	0000	0000	0011	0000	0000	0000	0000
Data	AA	X	AA	X	AA	CC	FF	X	FF	X	FF	KK	X	KK	X	MM
Ncopies	2	0	2	0	1	1	2	0	2	0	1	2	0	2	0	1

UNIVERSITY OF LIVERPOOL

Centre for Doctoral Training in DISTRIBUTED ALGORITHMS

Step 3: Fill the gaps (2/2)

Output from Step 3.2:

Data	AA	X	X	X	AA	CC	FF	X	X	X	FF	KK	X	X	X	MM
Ncopies	4	0	0	0	1	1	4	0	0	0	1	4	0	0	0	1

Step 3.3: Express Ncopies-1 in binary and split according to bit 2

Binary	0011	0000	0000	0000	0000	0000	0011	0000	0000	0000	0000	0011	0000	0000	0000	0000
Data	AA	X	AA	X	AA	CC	FF	X	FF	X	FF	KK	X	KK	X	MM
Ncopies	2	0	2	0	1	1	2	0	2	0	1	2	0	2	0	1

Step 3.3: Express Ncopies-1 in binary and split according to bit 1

Binary	0001	0000	0001	0000	0000	0000	0001	0000	0001	0000	0000	0001	0000	0001	0000	0000
Data	AA	AA	AA	AA	AA	CC	FF	FF	FF	FF	FF	KK	KK	KK	KK	MM
Ncopies	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

UNIVERSITY OF LIVERPOOL

Centre for Doctoral Training in DISTRIBUTED ALGORITHMS

Results

Comparison of run-time (and speed-up) as a function of hardware

- 8 machines, each mounting a 2 Xeon Gold 6138 CPU (32 (of 40) cores at 2 GHz) with InfiniBand 100 Gbps
- Proposed approach: Rotational Nearly Sort and Split (RoSS)
    - Uses some further optimisations (eg combines stages 2 and 3)
- Two pre-existing $O((\log_2 N)^2)$ baselines:
    - Bitonic-Sort-Based Redistribution (B-R)[6]
    - Nearly-Sort-Based Redistribution (N-R)[7]

Notation:

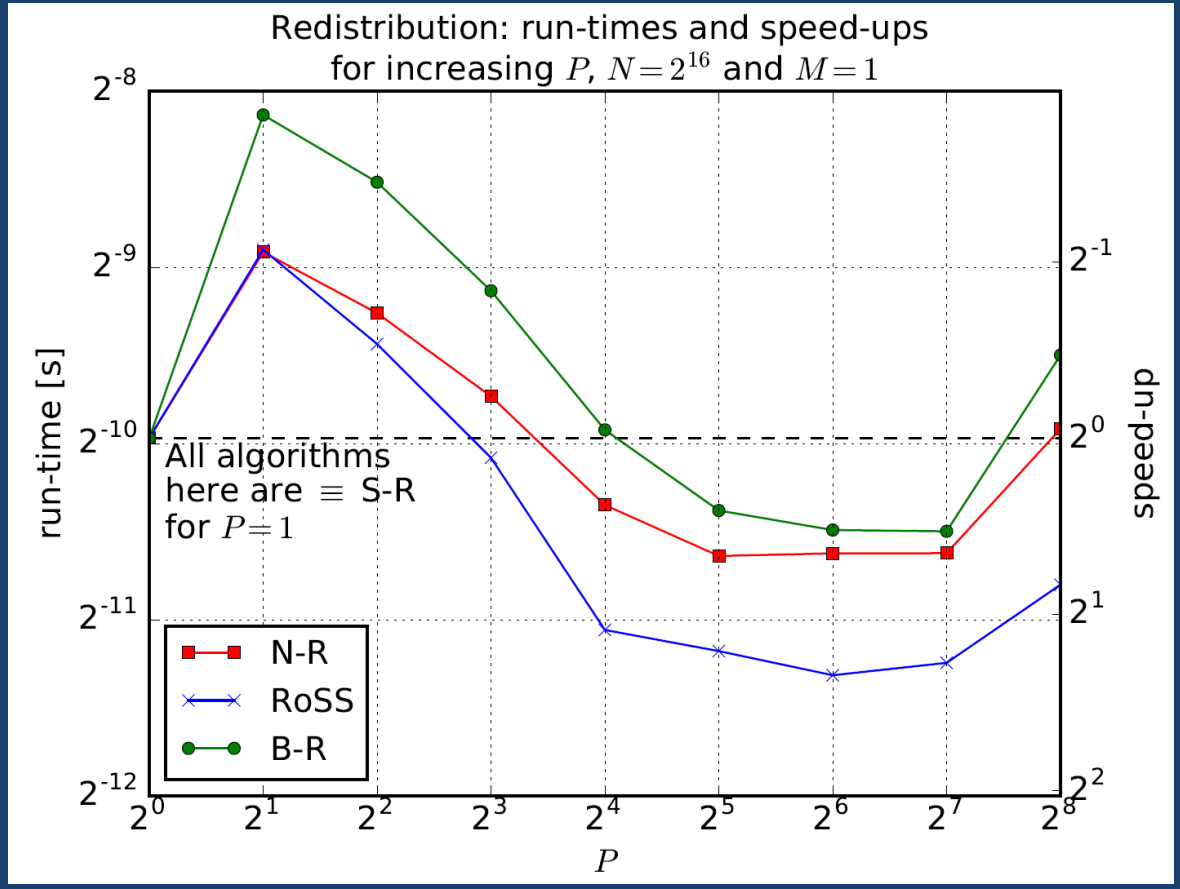
- P: number of processors (actually using multiple particles per node)
- N: number of samples (aka particles)
- T_{PF}: number of time-steps
- M: state's dimensionality (stochastic volatility (with bootstrap filter), to maximise challenge for resampling)

[6] Varsi, A.; Kekempanos, L.; Thiyagalingam, J.; Maskell, S. *Parallelising Particle Filters with Deterministic Runtime on Distributed Memory Systems*. IET 3rd International Conference on Intelligent Signal Processing (ISP 2017), London, UK, 4–5 December 2017; pp. 1–10
[7] A Varsi, L Kekempanos, J Thiyagalingam and S Maskell. A Single SMC Sampler on MPI that Outperforms a Single MCMC Sampler

UNIVERSITY OF LIVERPOOL

Centre for Doctoral Training in DISTRIBUTED ALGORITHMS

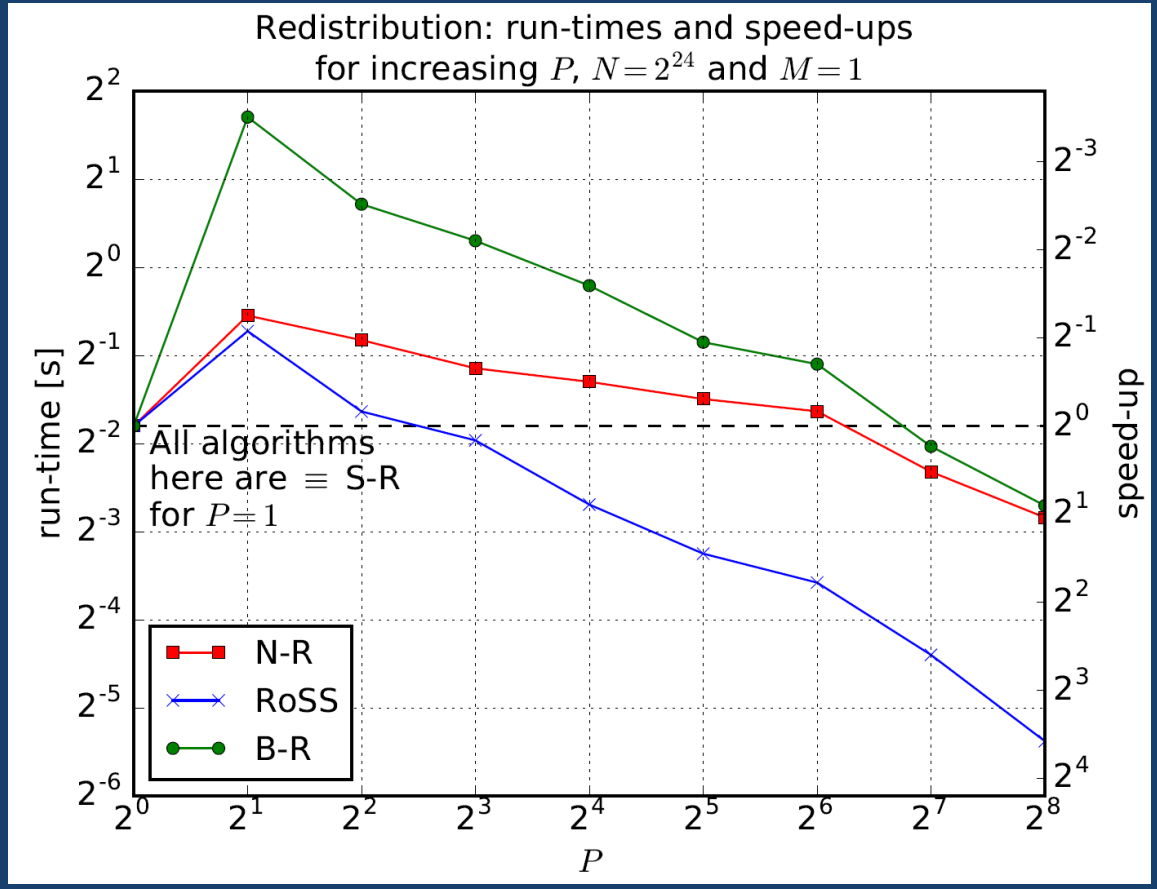
Results: Resampling



UNIVERSITY OF LIVERPOOL

Centre for Doctoral Training in DISTRIBUTED ALGORITHMS

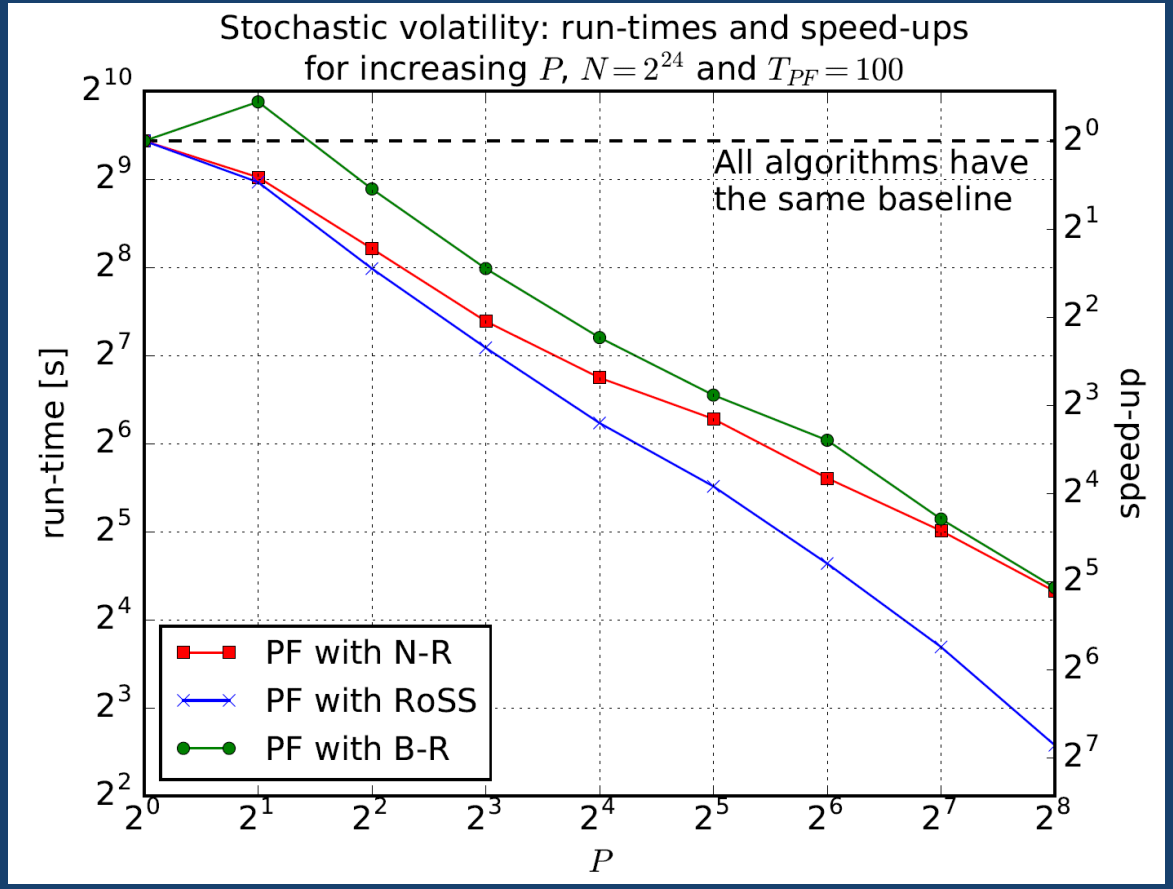
Results: Resampling



UNIVERSITY OF LIVERPOOL

Centre for Doctoral Training in DISTRIBUTED ALGORITHMS

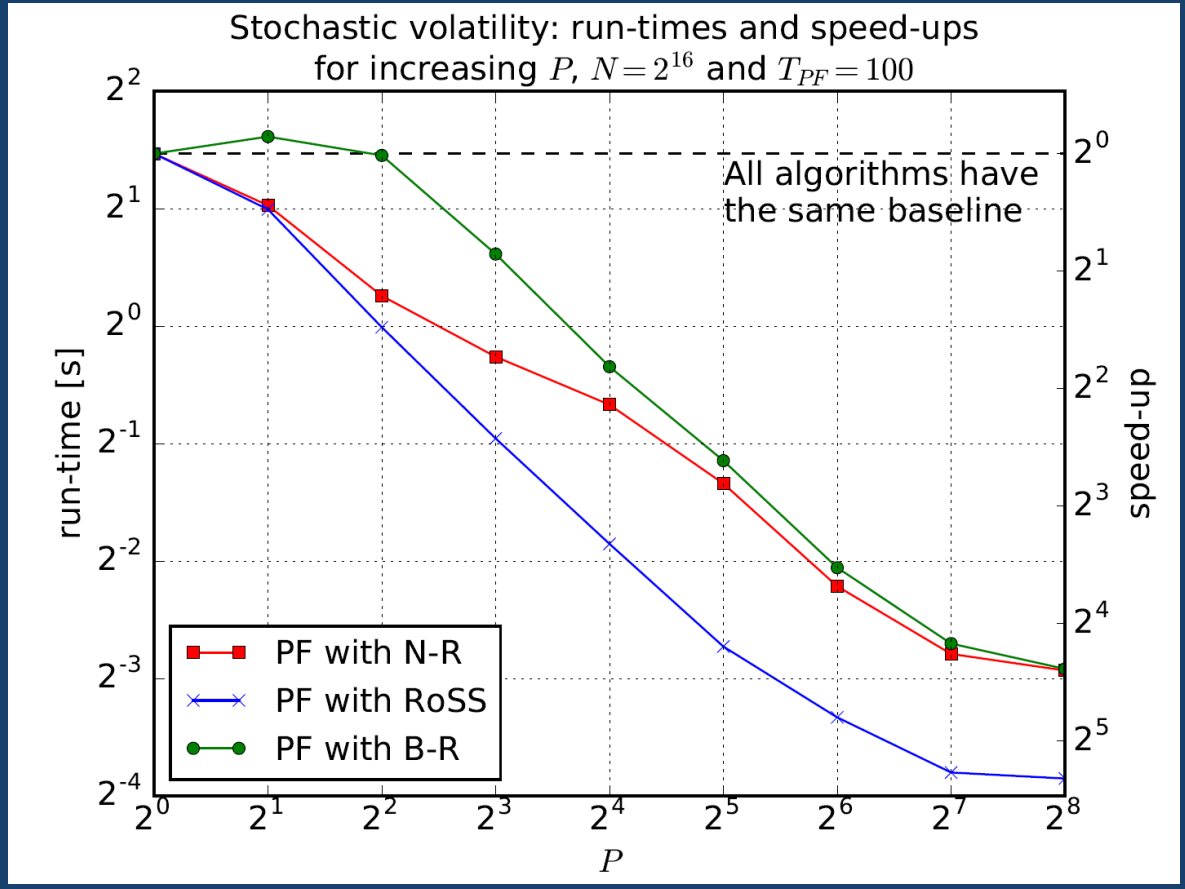
Results: A (Stressing) Exemplar Particle Filter



UNIVERSITY OF LIVERPOOL

Centre for Doctoral Training in DISTRIBUTED ALGORITHMS

Results: A (Stressing) Exemplar Particle Filter



UNIVERSITY OF LIVERPOOL

Centre for Doctoral Training in DISTRIBUTED ALGORITHMS

Conclusions

There exists and $O(\log_2 N)$ resampling algorithm for distributed memory systems

Results include a worst-case (!) 125x best speed-up with 256 cores

- Speed-ups will increase with more sophisticated proposal and model

UNIVERSITY OF LIVERPOOL

Centre for Doctoral Training in DISTRIBUTED ALGORITHMS

Future Work

Develop implementation

- Intersperse communications and computation
- Hybrid OpenMP/MPI version
- Quantify Performance across a range of hardware infrastructures and models

Fully capitalise on the technical capability that this advance facilitates

- Shift focus towards models and away from algorithms

Publish specific particle filter in Streaming-Stan

- A combination of FL-SMC and NUTS

Open source release of Streaming-Stan

- Adopt Stan's licencing terms

Integrate Streaming-Stan with Stan and SMC-Stan (on parameters using PMCMC/SMC^2)

- Eg for data-driven calibration of multi-target trackers

UNIVERSITY OF LIVERPOOL

Centre for Doctoral Training in DISTRIBUTED ALGORITHMS