High Performance Computing Cookbook: The BLonD Recipe.

A collection of HPC techniques to accelerate your scientific simulation workflow.

Konstantinos Iliakis
Doctoral Student
BE-RF-FB, CERN &
MicroLab, ECE, NTUA
konstantinos.iliakis@cern.ch

Supervisors:
Dr. Helga Timko, BE-RF-FB, CERN &
Prof. Dimitrios Soudris, ECE, NTUA
Advisor: Dr. Sotirios Xydis, DIT, HUA

Acknowledgements: The BLonD Dev team.

BE Seminars, May 29 2020, CERN, CH

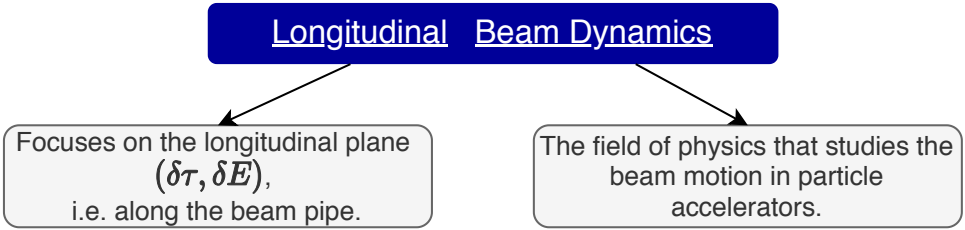
Table of Contents

1. Introduction & Background
   - Beam Dynamics & The BLonD Simulator
   - HPC Challenges & Motivation
2. Single-Node Optimizations
   - General Practices
   - BLonD++ Methodology
   - Evaluation & Summary
3. Multi-node Implementation
   - Distributed Computing
   - Base Execution Flow
   - Optimization Strategies
   - HBLonD Fine Tuning
   - Scalability Stressing
4. Conclusion

Table Of Contents

1. Introduction & Background
   - Beam Dynamics & The BLonD Simulator
   - HPC Challenges & Motivation
2. Single-Node Optimizations
   - General Practices
   - BLonD++ Methodology
   - Evaluation & Summary
3. Multi-node Implementation
   - Distributed Computing
   - Base Execution Flow
   - Optimization Strategies
   - HBLonD Fine Tuning
   - Scalability Stressing
4. Conclusion

Longitudinal Beam Dynamics and Simulations



Longitudinal Beam Dynamics and Simulations

Longitudinal Beam Dynamics

Focuses on the longitudinal plane $(\delta\tau, \delta E)$, i.e. along the beam pipe.

The field of physics that studies the beam motion in particle accelerators.

Why modelling beam evolution?

- Understand day-to-day observations.
- Fine tune and control the beam, ensuring *beam quality* and *safety*.
- Model and predict performance after upcoming upgrades.
- Experiment and design future machines.

Longitudinal Beam Dynamics and Simulations

Longitudinal Beam Dynamics

Focuses on the longitudinal plane $(\delta\tau, \delta E)$, i.e. along the beam pipe.

The field of physics that studies the beam motion in particle accelerators.

Why modelling beam evolution?

- Understand day-to-day observations.
- Fine tune and control the beam, ensuring *beam quality* and *safety*.
- Model and predict performance after upcoming upgrades.
- Experiment and design future machines.

Why developing a simulator?

- Highly complex machines, require numerical models.
- Maximum observability and flexibility.
- Limited resources devoted to beam studies (MDs).

The BLonD Simulator Suite

Origins

- Beam Longitudinal Dynamics (BLonD), specializes on the longitudinal plane.
- Written in Python, open-source and cross-platform (minimal dependencies).
- Modular structure, covers wide range of synchrotrons, energy regimes and particle types.
- Extends the functionality of older codes (e.g. ESME[1]).

[1] JA MacLachlan. *ESME: Longitudinal Phase Space Particle Tracking-Program Documentation*. Tech. rep. TM-1274. Fermi National Accelerator Lab.,

The BLonD Simulator Suite

Origins

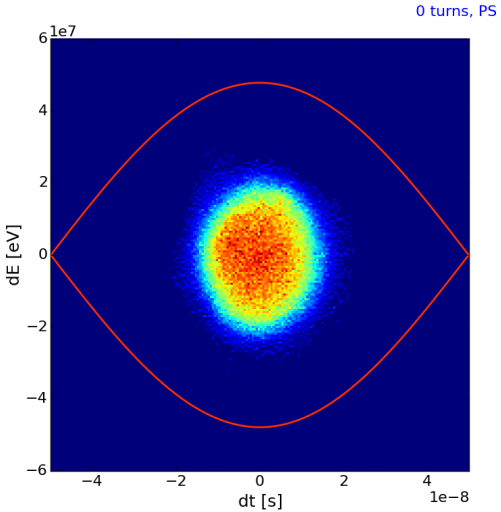
- Beam Longitudinal Dynamics (BLonD), specializes on the longitudinal plane.
- Written in Python, open-source and cross-platform (minimal dependencies).
- Modular structure, covers wide range of synchrotrons, energy regimes and particle types.
- Extends the functionality of older codes (e.g. ESME[1]).

Building Trust

- Benchmarking against theory, measurements, and other codes.
- Numerous published articles.
- Continuous development, testing, feature addition.
- Small but constantly growing user-base, at CERN, Fermilab, KEK, GSI and others.

[1] JA MacLachlan. *ESME: Longitudinal Phase Space Particle Tracking-Program Documentation*. Tech. rep. TM-1274. Fermi National Accelerator Lab.,

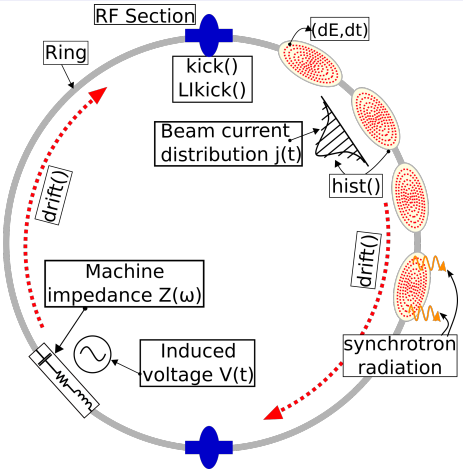
Example Application: PS-to-SPS Beam Transfer



- RF manipulations required for LHC-type, 25ns beam generation in the PS.
- Complex and precise operations such as splitting and bunch length shortening.
- Modeling beam evolution at injection, acceleration and flat-top.
- Cross-accelerator simulation capability.

The BLonD Simulator Architecture

- Main components: Beam, RF Sections, Ring.
- Particles in phase-space: $(\delta\tau_{(n)}, \delta E_{(n)})$
- **Equations of Motion**[2]: Energy change $\Rightarrow$ '**kick**' Motion between RF stations $\Rightarrow$ '**drift**' equation.
- A **beam cycle** in the ring $\Leftrightarrow$ **one step**.

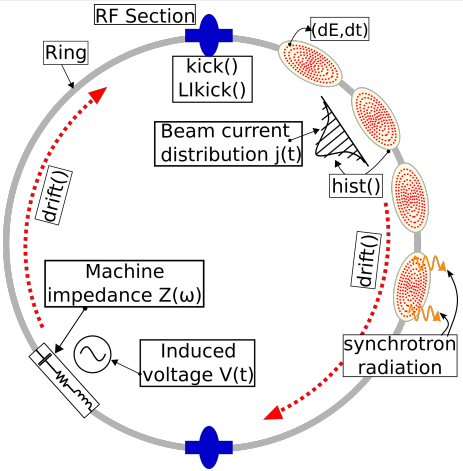


[2] *CERN Beam Longitudinal Dynamics code BLonD.* 2014. URL: `https://blond.web.cern.ch/` (visited on 03/02/2018)

[3] Helga Timko et al. "Benchmarking the beam longitudinal dynamics code BLonD". In: *Proceedings of the 7th International Particle Accelerator Conference (IPAC 2016): Busan, Korea.* 2016

The BLonD Simulator Architecture

- Main components: Beam, RF Sections, Ring.
- Particles in phase-space: $(\delta\tau_{(n)}, \delta E_{(n)})$
- **Equations of Motion**[2]: Energy change $\Rightarrow$ '**kick**' Motion between RF stations $\Rightarrow$ '**drift**' equation.
- A **beam cycle** in the ring $\Leftrightarrow$ **one step**.
- **Beam interacts with surroundings** $\Rightarrow V_{ind} = FFT(Z(\omega)S(\omega))$
- $V_{ind} \Rightarrow$ **Collective effects**, coupling the bunches.

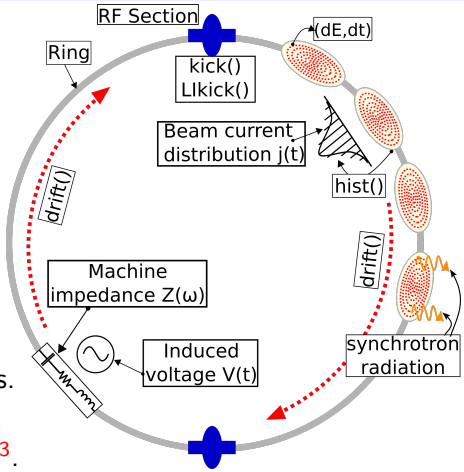


[2] *CERN Beam Longitudinal Dynamics code BLonD.* 2014. URL: `https://blond.web.cern.ch/` (visited on 03/02/2018)

[3] Helga Timko et al. "Benchmarking the beam longitudinal dynamics code BLonD". In: *Proceedings of the 7th International Particle Accelerator Conference (IPAC 2016): Busan, Korea.* 2016

The BLonD Simulator Architecture

- Main components: Beam, RF Sections, Ring.
- Particles in phase-space: $(\delta\tau_{(n)}, \delta E_{(n)})$
- **Equations of Motion**[2]: Energy change $\Rightarrow$ '**kick**' Motion between RF stations $\Rightarrow$ '**drift**' equation.
- A **beam cycle** in the ring $\Leftrightarrow$ **one step**.
- **Beam interacts with surroundings** $\Rightarrow V_{ind} = FFT(Z(\omega)S(\omega))$
- $V_{ind} \Rightarrow$ **Collective effects**, coupling the bunches.
- Profile of beam line density to optimize complexity.
- $\mathcal{O}(T \times (P + S \log S))$, T: turns, P: particles, S: slices.
- **Physics phenomena**, e.g. space-charge, synchrotron radiation, and intensity effects, **enabled on demand**[3].



[2] *CERN Beam Longitudinal Dynamics code BLonD.* 2014. URL: `https://blond.web.cern.ch/` (visited on 03/02/2018)

[3] Helga Timko et al. "Benchmarking the beam longitudinal dynamics code BLonD". In: *Proceedings of the 7th International Particle Accelerator Conference (IPAC 2016): Busan, Korea.* 2016

Beam Dynamics Challenges

Need for Excessive Studies and HPC Simulations

- **Technological and budgetary constraints**.
- **Operation** of **existing synchrotrons**.[4] [5] [6]
- **Planning** of the upcoming **machine upgrades**, e.g. LIU[7] and HL-LHC project[8].
- Evaluation of **future concepts**, e.g. FCC[9].

[4] Vincenzo Forte et al. "Longitudinal injection schemes for the CERN PS Booster at 160 MeV including space charge effects". In: *Procs. of the 6th IPAC in Richmond, VA, USA*. 2015

[5] Markus Schwarz et al. "Flat-Bottom Instabilities in the CERN SPS". In: *10th Int. Partile Accelerator Conf.(IPAC'19), Melbourne, Australia*. JACOW. 2019, pp. 3224–3227

[6] Joël Repond et al. "Simulations of Longitudinal Beam Stabilisation in the CERN SPS With BLonD". In: *Proceedings ICAP2018: Key West, FL, USA*. 2018, TUPAF06

[7] Danilo Quartullo, Simon Albright, Elena Shaposhnikova, et al. "Studies of Longitudinal Beam Stability in CERN PS Booster After Upgrade". In: *IPAC'17, Copenhagen, Denmark, May, 2017*. JACOW. 2017

[8] Apollinari G. et al. *High-Luminosity Large Hadron Collider (HL-LHC): Technical Design Report V. 0.1*. CERN Yellow Reports: Monographs. Geneva: CERN, 2017. URL: http://cds.cern.ch/record/2284929

[9] "The Future Circular Collider study". In: *CERN Courier* 54.3 (2014), pp. 16–18. URL: http://cds.cern.ch/record/2064538

Workload Characteristics

High-energy particles, circulating inside accelerators

- Scientifically and computationally challenging task.
- Inherently parallel, natural fit for distributed environment.
- Intra-node scaling and shared-memory usage is essential.
- Fine tuning and optimization of scientific codes is a complex, manual endeavor.
- Awareness of the underlying architecture is crucial.

Contributions

To undertake these challenges:

1. Comprehensive methodology to optimize the single-node performance of BLonD $\Rightarrow$ **BLonD++**.
2. We use message passing to enable distributed computing and multi-node scaling $\Rightarrow$ **HBLonD**.
3. Exploit **mixed** data- and task-**parallelism**.
4. **Optimize the network traffic** to minimize communication overhead.
5. Consider **floating point size** performance-accuracy **tradeoff**.
6. Build a **Dynamic Load Balancing run-time** to coordinate all the above.
7. Demonstrate a minimum of **two orders of magnitude reduced runtime**, week-long simulations in a couple of hours.

Table Of Contents

1. Introduction & Background
   - Beam Dynamics & The BLonD Simulator
   - HPC Challenges & Motivation
2. Single-Node Optimizations
   - General Practices
   - BLonD++ Methodology
   - Evaluation & Summary
3. Multi-node Implementation
   - Distributed Computing
   - Base Execution Flow
   - Optimization Strategies
   - HBLonD Fine Tuning
   - Scalability Stressing
4. Conclusion

Software Engineering (SWE) and Tooling (I)

Equip your application with SWE practices early on.

- Unit-testing can be tedious, but will pay off quickly.
- Continuous integration and testing: Plethora of tools to choose from.
- Ensure easy, portable installation in separate environments: Simplifies development and user adoption.

Programming Language

- Performance oriented.
- Low-level control: Memory management, thread/ process management, concurrency.
- Backed-up by a body of HPC libraries, parallel frameworks.
- Usually, C and C++ are top contenders.

Software Engineering (SWE) and Tooling (II)

Familiarize yourself with a set of essential tools.

- A debugger, e.g. GDB, PDB, Valgrind, Strace, etc.
- Compiler: Depends on the language, the supported platforms, a combination may be preferable.
- Time profiling tools: Large variety, e.g. Linux Perf, Intel VTune, language specific libraries.
  - How much time did this optimization save?
  - What are the bottlenecks of my code?
  - How well does a region scale?
- Performance Counters: Perf, VTune, PAPI.
  - Why is this region performing sub-optimally?
  - When should I stop optimizing?
  - Where in my processor is the bottleneck? The memory, the CPU, something else?

Intra-node Optimizations

Low-hanging fruits

- Compiler and compiler optimization flags.
  - Be careful, some are "unsafe".
- Use 3rd party, optimized libraries, e.g. VDT[10], Intel MKL[11], FFTW[12], Boost[13].
  - No need to reinvent the wheel every time.
- Keep in mind: *Premature Optimization is the root of all evil.*

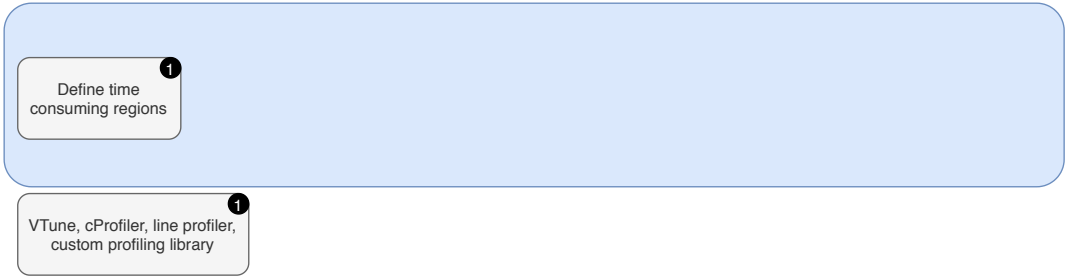
[10] Danilo Piparo, Vincenzo Innocente, and Thomas Hauth. "Speeding up HEP experiment software with a library of fast and auto-vectorisable mathematical functions". In: *Journal of Physics: Conference Series* (2014)

[11] *Intel® Math Kernel Library*. 2018. URL: https://software.intel.com/en-us/mkl

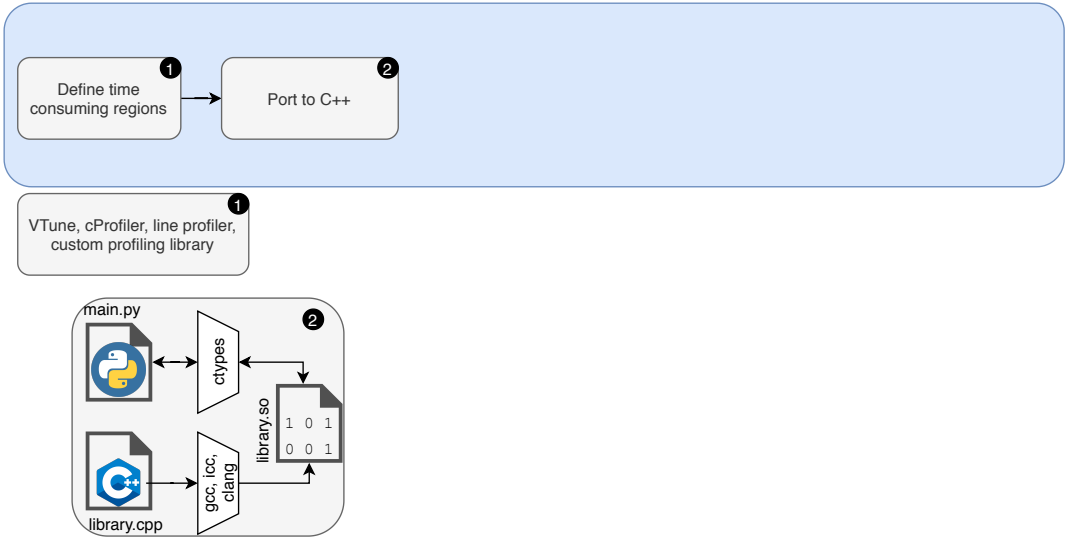
[12] Matteo Frigo and Steven G Johnson. "FFTW: An adaptive software architecture for the FFT". In: *Procs. of the 1998 Int. Conf. on Acoustics, Speech and Signal Processing, ICASSP'98*. IEEE. 1998

[13] David Abrahams and Aleksey Gurtovoy. *C++ Template Metaprogramming: Concepts, Tools, and Techniques from Boost and Beyond (C++ in Depth Series)*. Addison-Wesley Professional, 2004

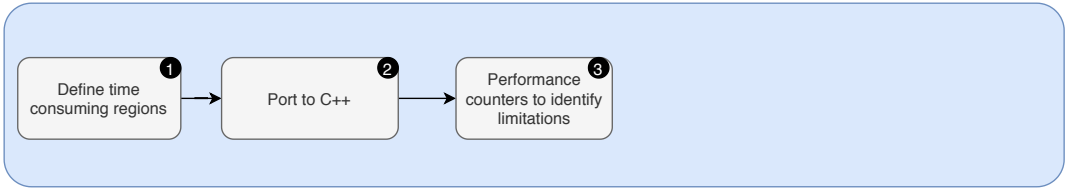
Single-node Optimization Strategy



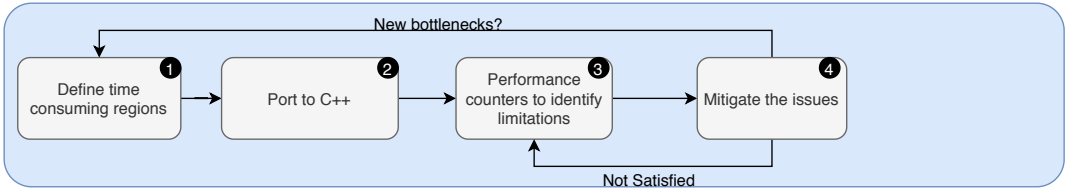
Single-node Optimization Strategy



Single-node Optimization Strategy

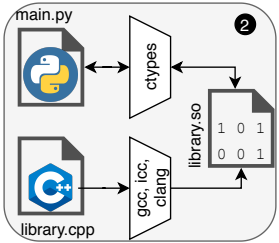


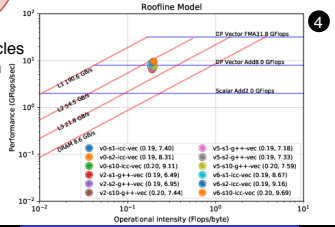
Single-node Optimization Strategy



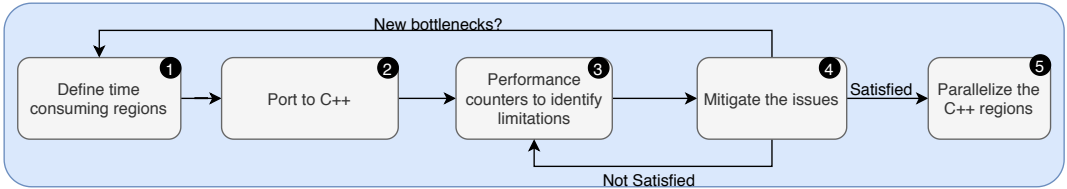
1. VTune, cProfiler, line profiler, custom profiling library

4. data structures, algorithms, vectorization, 3rd party libraries, tiling etc





Single-node Optimization Strategy

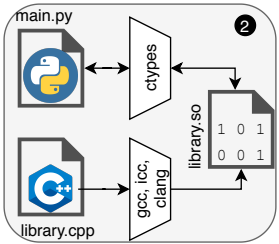


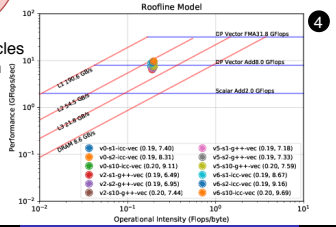
1. VTune, cProfiler, line profiler, custom profiling library

3. Processor Cycles Breakdown

4. data structures, algorithms, vectorization, 3rd party libraries, tiling etc

5. OpenMP, MPI, Pthreads, C++ threads

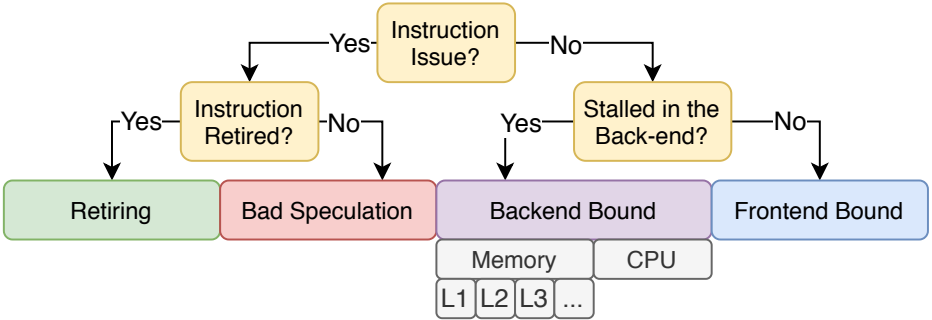




Performance Counters to Discover Bottlenecks

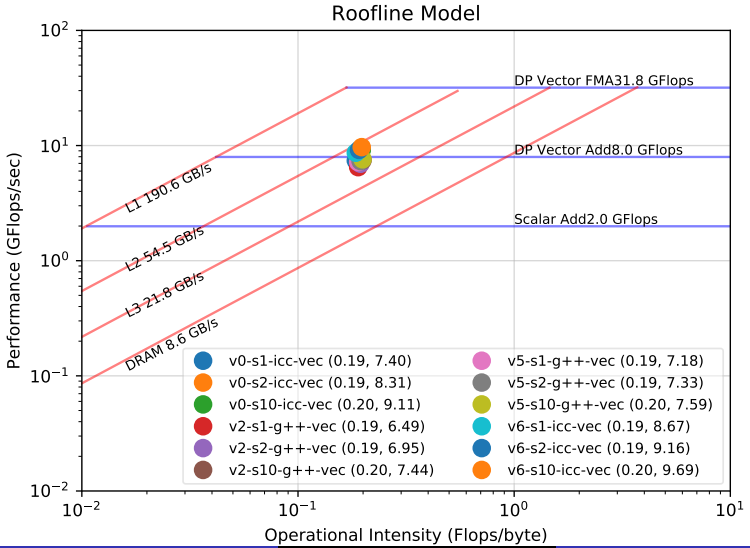
Top-down Micro-architecture Analysis Method (TMAM) [14]

- Based on the processor's performance monitoring units.
- Hierarchical breakdown of the processor's pipeline slots into various categories.
- Generic, low-overhead and offers clear insights on performance bottlenecks.



[14] Ahmad Yasin. 2014. A top-down method for performance analysis and counters architecture. In ISPASS, 2014, IEEE

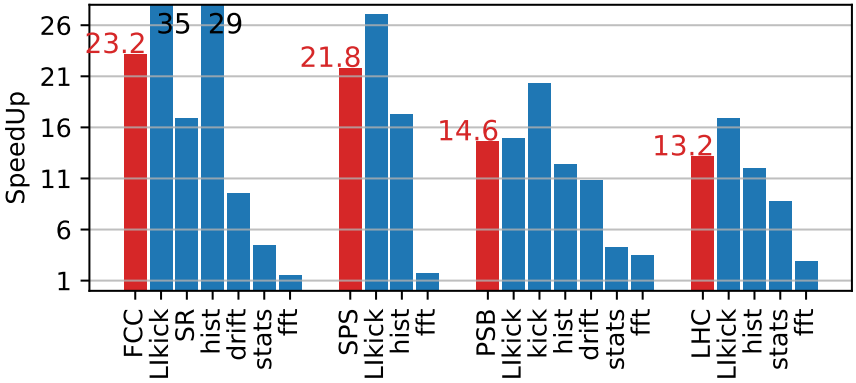
The Roofline Model



- A simple, intuitive way to estimate the potential of an algorithm run in an architecture.
- X-axis: Arithmetic intensity, property of the algorithm.
  Y-axis: Throughput in Flops/sec.
- Perform associated optimizations to break through the ceilings.
- More detail: Samuel Williams, Andrew Waterman, and David Patterson. "Roofline: an insightful visual performance model for multicore architectures". In: *Communications of the ACM* 52.4 (2009), pp. 65–76
  and
  Aleksandar Ilic, Frederico Pratas, and Leonel Sousa. "Cache-aware roofline model: Upgrading the loft". In: *IEEE Computer Architecture Letters* 13.1 (2013), pp. 21–24

BLonD++ single-core cumulative speedup

- Single-core cumulative speedup of *BLonD++* over the initial *BLonD* version in 4 representative use-cases.
- 7 methods were responsible for 99% of the total run-time.
- *BLonD++* is 18× faster on average.



Intra-Node Optimizations Summary

BLonD++[15] Summary

- Python front-end with **C++ computational core**.
- Effective **SIMD** vector unit usage, Multi-Threaded (**OpenMP**).
- **High-performance** third-party **libraries**, e.g. Intel MKL[16], VDT[17], FFTW[18].
- Top-Down method for performance analysis (**TMAM**) to **identify and mitigate uarch bottlenecks**.
- Roofline model to evaluate the throughput w.r.t. the upper bound.
- On average **18x faster** than the original code version.

[15] Konstantinos Iliakis et al. "BLonD++: performance analysis and optimizations for enabling complex, accurate and fast beam dynamics studies". In: *18th Int. Conf. on Embedded Computer Systems: Architectures, Modeling, and Simulation, Pythagorion, Greece.* 2018. URL: https://doi.org/10.1145/3229631.3229640

[16] *Intel® Math Kernel Library.* 2018. URL: https://software.intel.com/en-us/mkl

[17] Danilo Piparo, Vincenzo Innocente, and Thomas Hauth. "Speeding up HEP experiment software with a library of fast and auto-vectorisable mathematical functions". In: *Journal of Physics: Conference Series* (2014)

[18] Matteo Frigo and Steven G Johnson. "FFTW: An adaptive software architecture for the FFT". In: *Procs. of the 1998 Int. Conf. on Acoustics, Speech and Signal Processing, ICASSP'98.* IEEE. 1998

Table Of Contents

1. Introduction & Background
   - Beam Dynamics & The BLonD Simulator
   - HPC Challenges & Motivation
2. Single-Node Optimizations
   - General Practices
   - BLonD++ Methodology
   - Evaluation & Summary
3. Multi-node Implementation
   - Distributed Computing
   - Base Execution Flow
   - Optimization Strategies
   - HBLonD Fine Tuning
   - Scalability Stressing
4. Conclusion

Current Limitations and Future Requirements

Anticipating Future Requirements

- BLonD++ memory bounded when increasing the thread count: The CPU data processing rate exceeds the memory data transfer rate.
- New simulation scenarios push for ever growing input sizes.
- Scaling within the processor boundaries (vertical) neither practical nor feasible.
- Need for multi-node scaling (horizontal).

Current Limitations and Future Requirements

Anticipating Future Requirements

- BLonD++ memory bounded when increasing the thread count: The CPU data processing rate exceeds the memory data transfer rate.
- New simulation scenarios push for ever growing input sizes.
- Scaling within the processor boundaries (vertical) neither practical nor feasible.
- Need for multi-node scaling (horizontal).

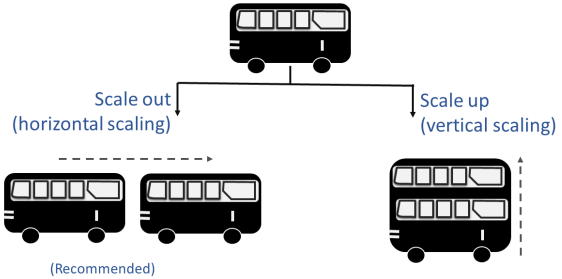


Figure 1: Horizontal versus vertical scaling. Credit: www.bmc.com.

Emergence of Hybrid-BLonD

Distributed Computing

- Multiple computing nodes *cooperatively* running a BLonD experiment.
- Cooperation $\Leftrightarrow$ Communication. Exchange of messages over the network.
- MPI: Message Passing Interface.

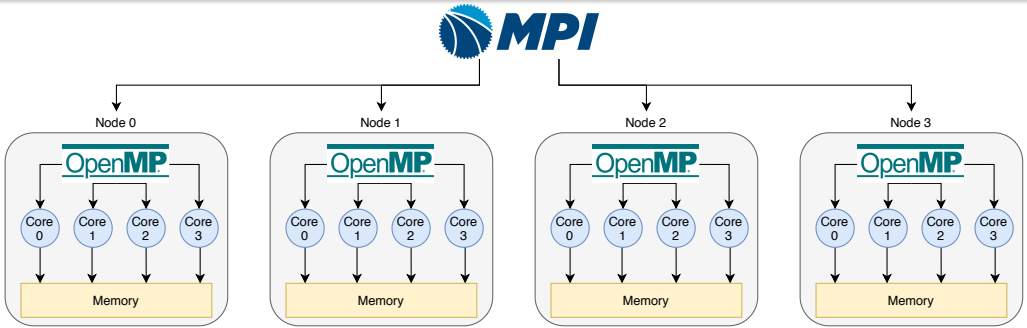


Figure 2: The Hybrid-BLonD MPI-over-OpenMP runtime.

HBLonD Overview

- Target: Computation, Minimize: Communication, Intra-worker processing.

HBLonD Overview

- Target: Computation, Minimize: Communication, Intra-worker processing.

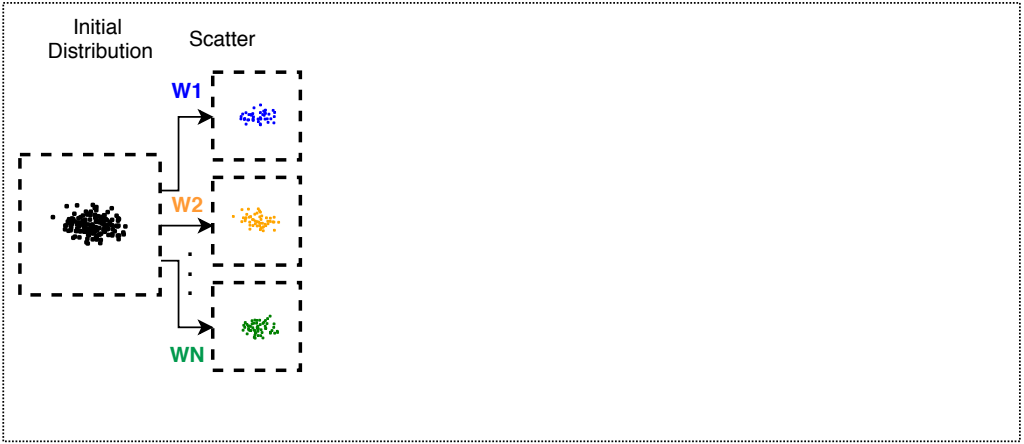


Figure 3: Baseline Hybrid-BLonD work-flow.

HBLonD Overview

- Target: Computation, Minimize: Communication, Intra-worker processing.

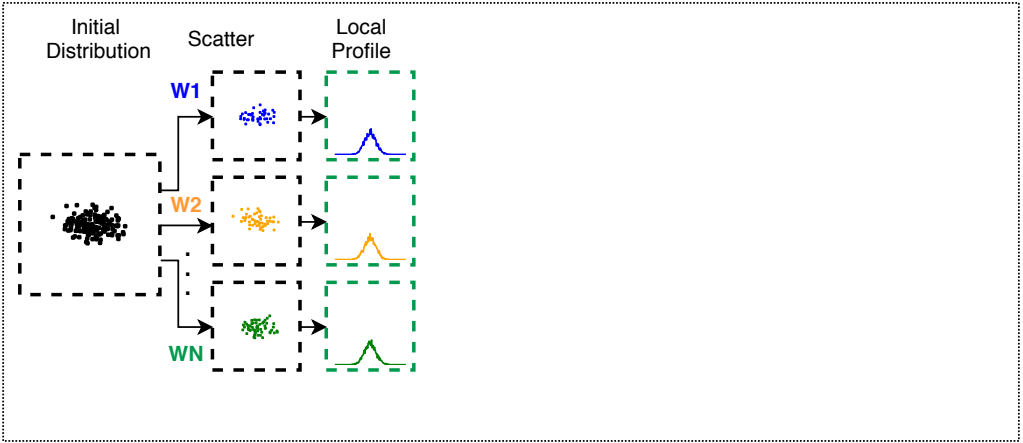


Figure 3: Baseline Hybrid-BLonD work-flow.

HBLonD Overview

- Target: Computation, Minimize: Communication, Intra-worker processing.

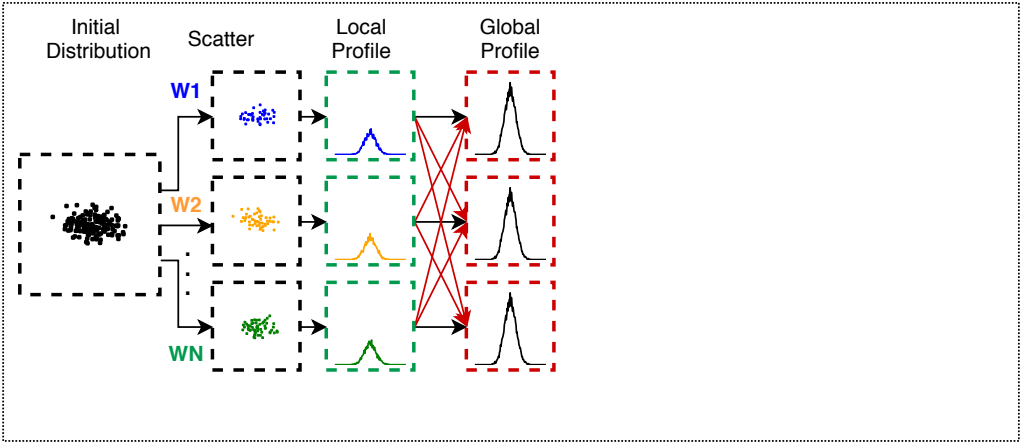


Figure 3: Baseline Hybrid-BLonD work-flow.

HBLonD Overview

- Target: Computation, Minimize: Communication, Intra-worker processing.

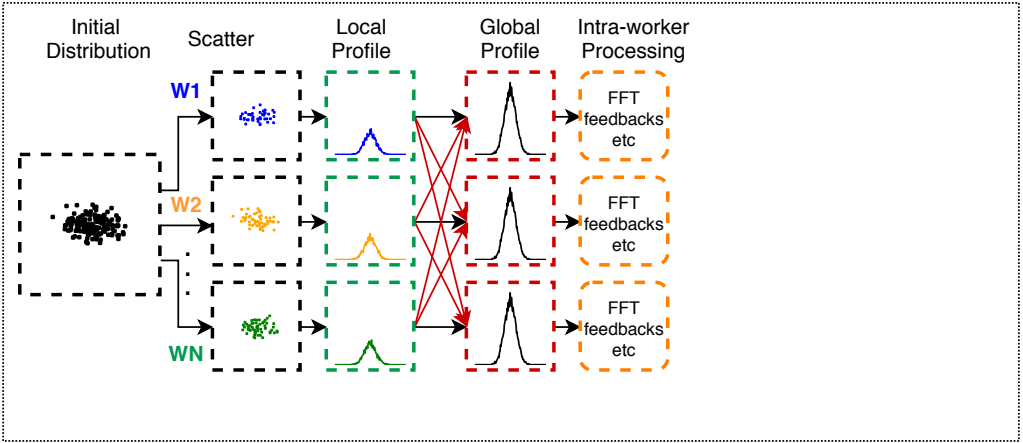


Figure 3: Baseline Hybrid-BLonD work-flow.

HBLonD Overview

- Target: Computation, Minimize: Communication, Intra-worker processing.

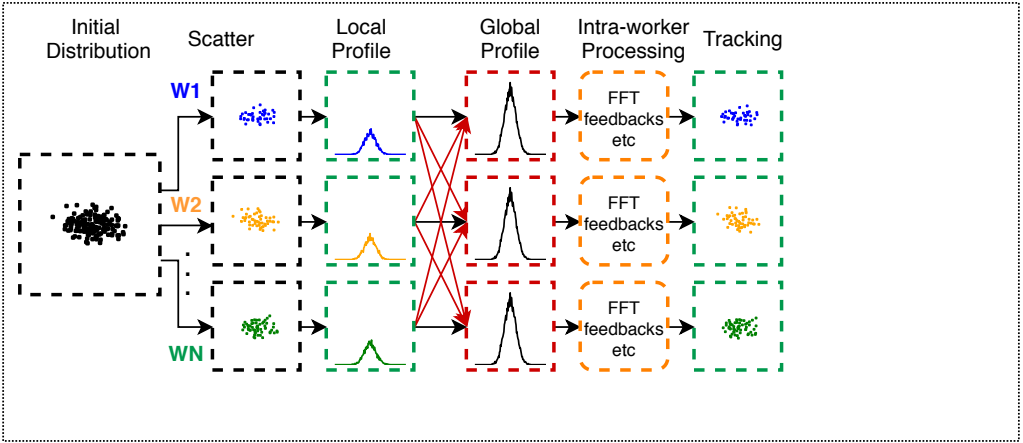


Figure 3: Baseline Hybrid-BLonD work-flow.

HBLonD Overview

- Target: Computation, Minimize: Communication, Intra-worker processing.

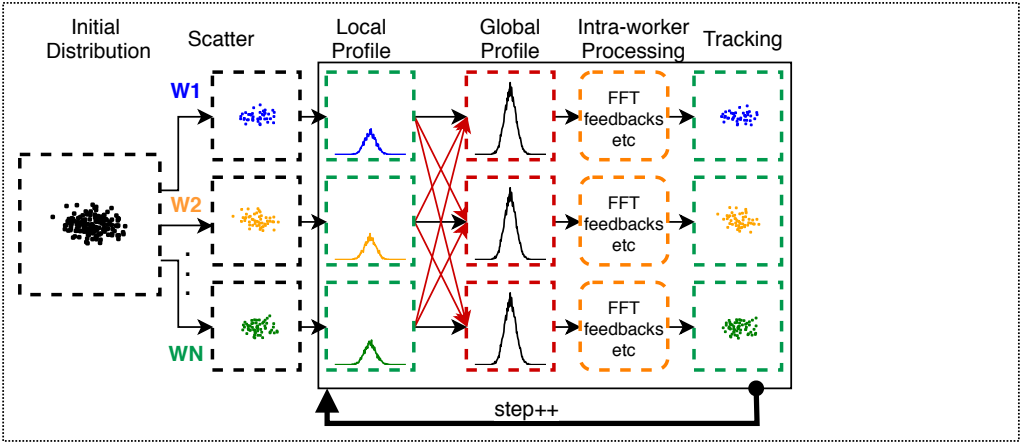


Figure 3: Baseline Hybrid-BLonD work-flow.

HBLonD Overview

- Target: Computation, Minimize: Communication, Intra-worker processing.

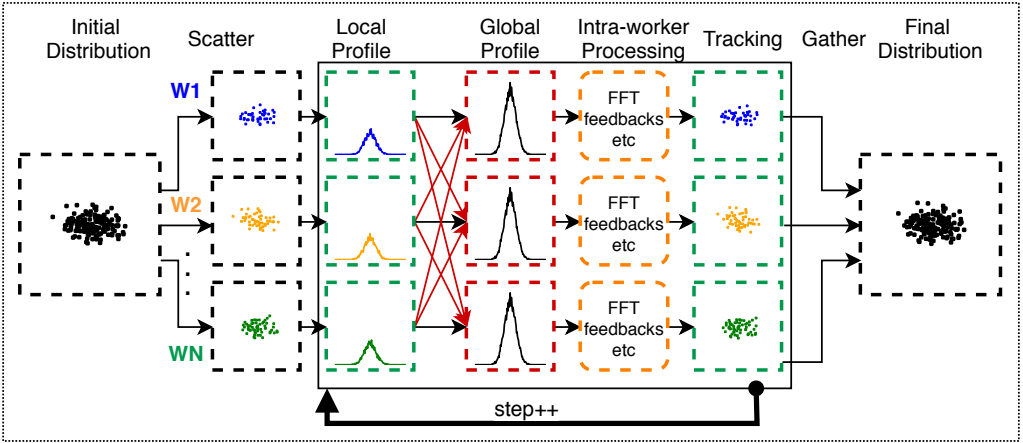


Figure 3: Baseline Hybrid-BLonD work-flow.

Mixed Data and Task-Parallelism

Types of Parallelism

- Data Parallelism: Same operations, different dataset.
- Task Parallelism: Different operations, same dataset.

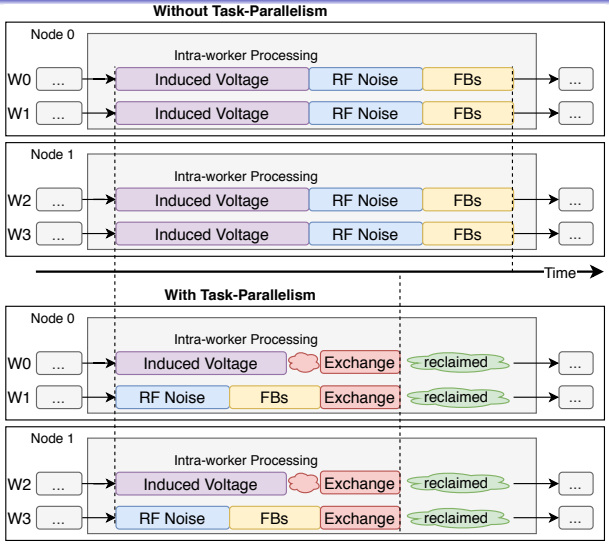
Mixed Data and Task-Parallelism

Types of Parallelism

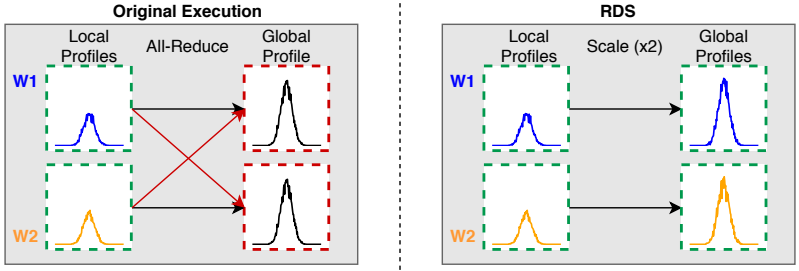
- Data Parallelism: Same operations, different dataset.
- Task Parallelism: Different operations, same dataset.

Mixed Parallelism

- Data-parallelism across workers.
- Task parallelism in the intra-worker processing stage.
- Results exchanged through fast shared memory.
- Imbalanced workload ⇒ lost time.



Network Traffic Optimization: RDS



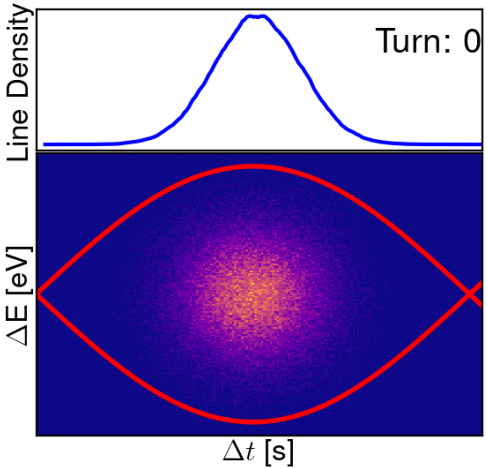
Representative Distribution Subset (**RDS**)

- Subset of particles assigned to a worker, form a representative subset of the whole distribution.
- Scale-up local profile to approximate the global profile.
- Avoids costly all-to-all reduction.
- Relaxes communication of MPI workers.

Network Traffic Optimization: SRP

Smoothly Revolving Profile (**SRP**)

- The beam profile is not rapidly changing between consecutive turns.
- Calculate it periodically instead of every turn.
- Thus, avoid per turn profile calculation, and other operations, consequently.



Floating Point Datatype Length

Single versus Double Floating Point Precision

- Single precision (f32): 32 bits long, 6-9 significant digits.
- Double precision (f64): 64 bits long, 15-17 significant digits.
- Basic accuracy/ performance trade-off.
- F32 numerical error less significant than statistical, discretization and other algorithmic errors.

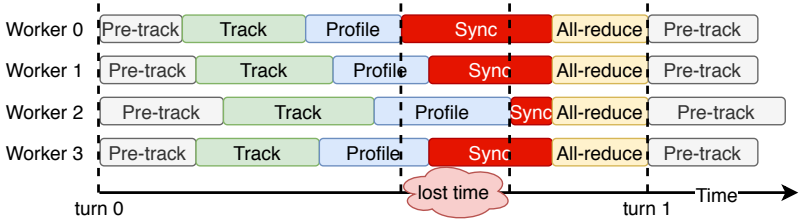
```
1 #include <math.h>
2 float num = 64.0;
3 float result = sqrt(num); // num casted to double
4 result = sqrtf(num); // invokes the 32-bit sqrt
```

Be careful: Type-casting

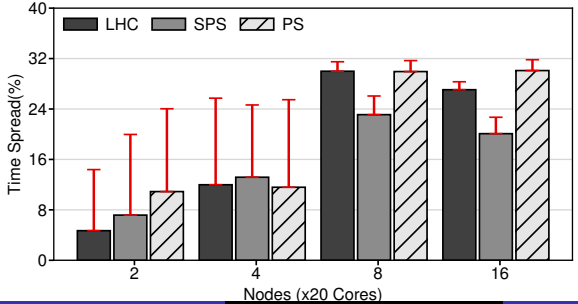
Table 1: f32/f64 reduced run-time.

	LHC	SPS	PS
Track (%)	37.4	42.4	54.7
Profile (%)	17.5	25.0	18.6
Intensity (%)	50.0	57.6	49.1
Total (%)	**31.7**	**47.3**	**33.2**

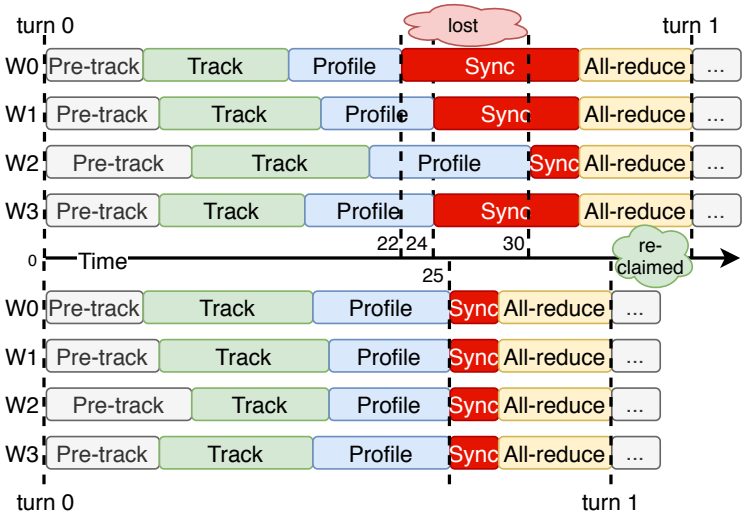
Imbalanced Execution



- Observed **spontaneous delay increase** of some nodes (**up to 30%**).
- $\Rightarrow$ Imbalanced execution.
- Cluster-derived, not data related.
- Slowest worker defines the turn time.



Dynamic Load-Balancing Scheme



Assumptions

1. $T_{comp} \propto P$.
2. Constant T_{comm} and T_{intra}.
3. Perfect LB $\Leftrightarrow T_{sync} \rightarrow min$.
4. Workers exhibit similar behavior for long periods.

Features

1. Triggered periodically, adaptively.
2. Minimizes transactions, prioritizes intra-node.
3. Progressively improving.

MPI Implementation and Workers-Per-Node Tuning

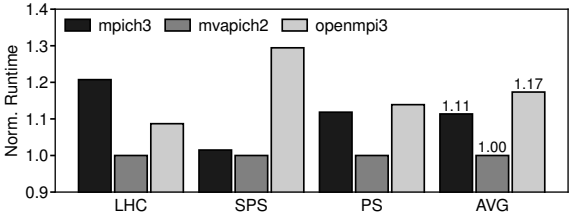


Figure 4: MPI implementation benchmark

- Tested OPENMPI3, MPICH3 and MVAPICH2.
- **MVAPICH2** 11% and 17% **faster** than MPICH3 and OPENMPI3, respectively.
- MVAPICH2 also the **most stable**.

MPI Implementation and Workers-Per-Node Tuning

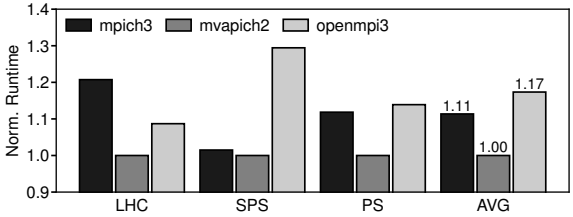


Figure 4: MPI implementation benchmark

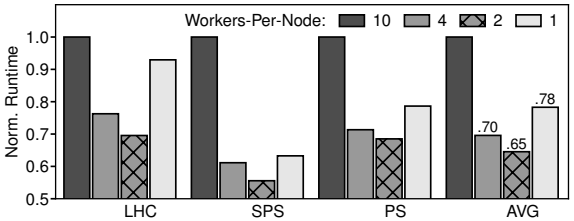
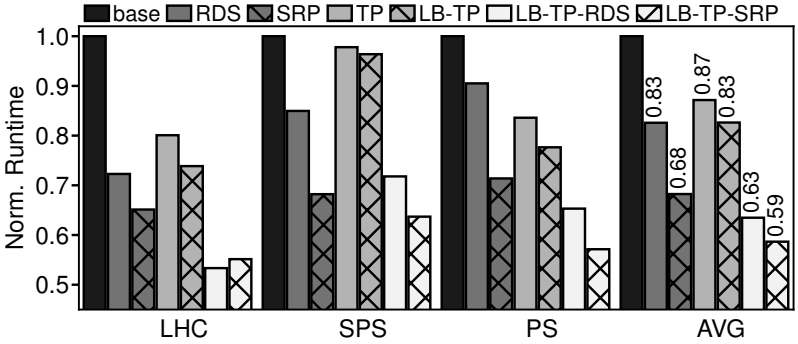


Figure 5: Worker-per-node tuning.

- Tested OPENMPI3, MPICH3 and MVAPICH2.
- **MVAPICH2** 11% and 17% **faster** than MPICH3 and OPENMPI3, respectively.
- MVAPICH2 also the **most stable**.

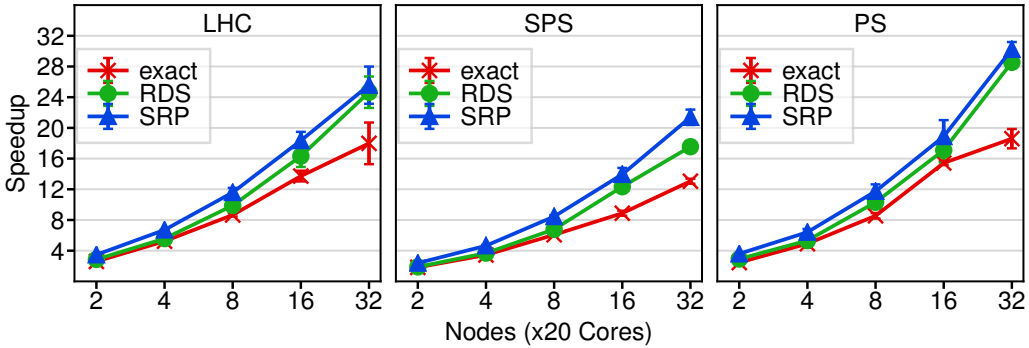
- Best performance with 1 worker per socket (NUMA domain).
- Cluster composed of dual-socket servers.
- Thus, **2 workers-per-node** best configuration.

Intermediate Effect Analysis



- The effect of the various optimizations on the execution time.
- **+17%** on average with Task-Parallelism and DLB.
- **+37%** and **+41%** on average with network traffic reduction.
- Impressive results, considering heavily optimized baseline.

Strong Scaling Performance Evaluation



- Constant input size, increasing number of nodes, up to **32 nodes, 640 cores**.
- *exact*: With DLB and mixed parallelism. *SRP* and *RDS*: With traffic optimization.
- Near optimal scalability up to 16 nodes. **Smaller scale simulations matter too.**
- Maximum average speedup **25.7x** (with SRP). Up to **30x** speedup (PS case).

Runtime Estimation and User Experience

Table 2: Real-World Case-Study Runtime Estimation.

Case	Turns	Particles	Bunches	*BLonD++*	*HBLonD*	Speedup
LHC	14M	$4 \cdot 10^6$	192	110 days	4.3 days	25.6×
SPS	430K	$6 \cdot 10^6$	288	8.8 days	9.8 hours	21.4×
PS	380K	$32 \cdot 10^6$	21	2.3 days	110 mins	30.2×

- Studies composed of **1000s of runs**.
- Run times $>$ 1 week, not practical.
- **LHC and SPS cases prohibitive**.
- **HBLonD enables** scientists to simulate and discover multi-bunched effects, **impossible to observe in the past**.

Table Of Contents

1. Introduction & Background
   - Beam Dynamics & The BLonD Simulator
   - HPC Challenges & Motivation
2. Single-Node Optimizations
   - General Practices
   - BLonD++ Methodology
   - Evaluation & Summary
3. Multi-node Implementation
   - Distributed Computing
   - Base Execution Flow
   - Optimization Strategies
   - HBLonD Fine Tuning
   - Scalability Stressing
4. Conclusion

Future Work

Main Extensions

1. More intelligent DLB scheme.
   - Re-balancing interval.
   - Predictive mechanism for early detection of emerging imbalances.
2. GPU based accelerated version of HBLonD (MPI-over-CUDA).
   - Harnessing the throughput of GPUs to accelerate the computational core of HBLonD.
3. Heterogeneous computing: combine CPU and GPU simultaneous, multi-node processing.
   - Using the DLB scheme to distribute the workload in a heterogeneous environment.

Conclusions

Demonstrated a series of HPC recipes

- Following a **comprehensive methodology**, and
- targeting the **single-node and multi-node performance**.
- Most of them, **can be adapted to the needs of other scientific simulators**, e.g. DLB, mixed-parallelism, floating point precision trade-off.
- A minimum **two orders of magnitude speedup**, w.r.t. the original BLonD code.

Next Generation BLonD Simulations

- Week long simulations down to a few hours.
- 288 SPS bunches brought light to effects previously unseen; crucial for LIU.
- HBLonD **made possible** novel simulation scenarios, **essential in the field** of longitudinal beam dynamics, that **help research advance**.

Thank you for your attention!

Konstantinos Iliakis konstantinos.iliakis@cern.ch
Helga Timko helga.timko@cern.ch
Sotirios Xydis sxydis@hua.gr
Dimitrios Soudris dsoudris@microlab.ntua.gr

The BLonD Dev Team:

S. Albright, T. Argyropoulos, H. Damerau, I. Karpov, A. Lasheen, L. E. Medina Medrano, J. E. Muller, G. Papotti, D. Quartullo, J. Repond, M. Schwarz, P. Tsapatsaris, M. Vadai
Wolfgang Hofle and Elena Shaposhnikova

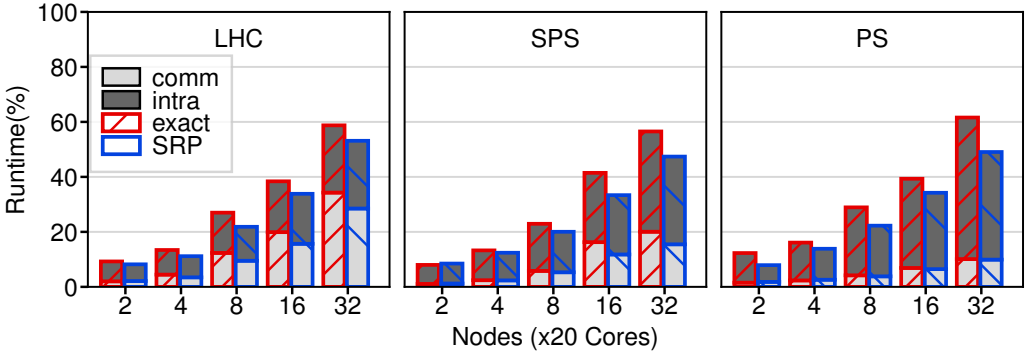
BLonD++: SAMOS 2018 Conference
HBLonD: DATE 2020
HBLonD: Computing Frontiers 2020
HBLonD: Artifact Persistent Repository

BE Seminars, May 29 2020, CERN, CH

Table Of Contents

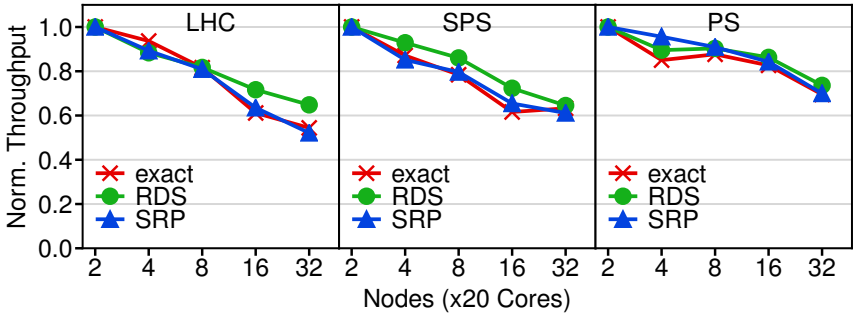
5 Appendix

Run Time Breakdown



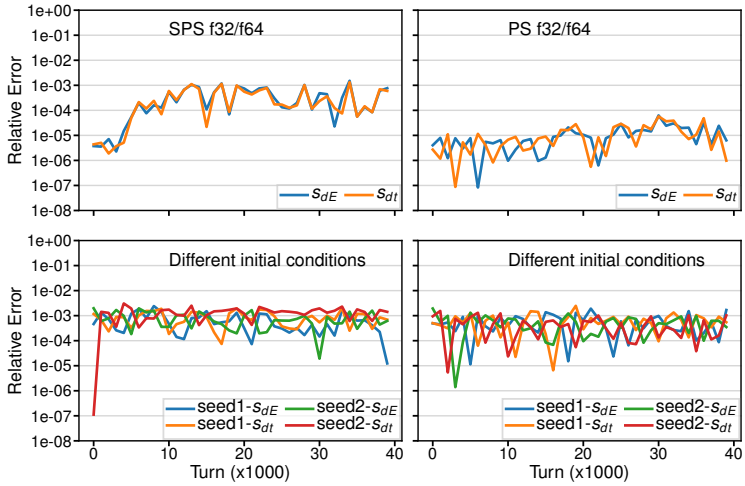
- **Runtime Breakdown** in communication, intra-node processing, and computation.
- Ideally: **Minimum** time spend in **comm** and **intra**.
- Node count increase: run time **dominated** by **communication and intra-worker processing stage**.

Weak Scaling Experiment



- Constant workload per node.
- Especially important for **smaller scale** simulations.
- Needed when running a large **design space exploration**.

Single vs Double Precision Evaluation



- Relative error: $|1 - \frac{value_{f32}}{value_{f64}}|$
- Initial conditions: Seed value used for input generation.
- F32 numerical error less significant than statistical error.

Table 3: f32/f64 reduced run-time.

	LHC	SPS	PS
Track (%)	37.4	42.4	54.7
Profile (%)	17.5	25.0	18.6
Intensity (%)	50.0	57.6	49.1
Total (%)	**31.7**	**47.3**	**33.2**