Distributed Optimisation with Evolutionary Algorithms

Geneva with an MPI Consumer

Kilian Schwarz, Matthias Lutz, Jonas Wessner, Rüdiger Berlich
DESY, June 6, 2023

HELMHOLTZ

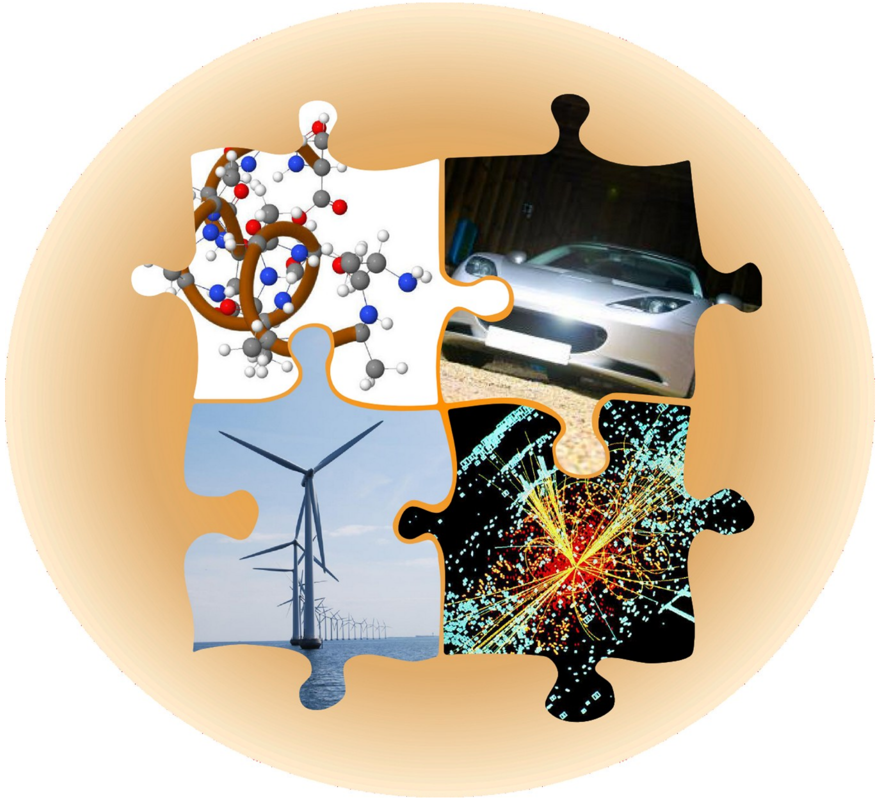
DESY.

Agenda

Introduction into Geneva

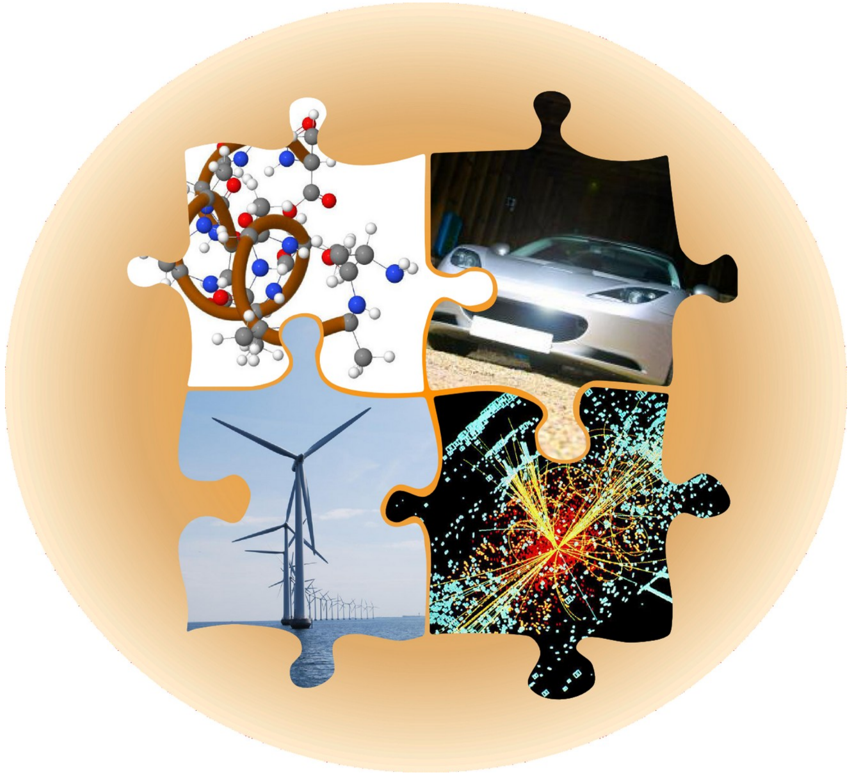
The Geneva Library Collection

- Generic C++ framework for the search for **optimised solutions** of technical and scientific problems
- Covering **Evolutionary Algorithms**, Swarm Algorithms, Simulated Annealing, Parameter Scans and Gradient Descents
- Data structures allow direct interaction between different optimisation algorithms with **just one problem description**
- **Inter-parameter constraints (x+y < 1) possible**
- Support for **many-core systems** as well as parallel and distributed environments
- Available under the Apache License v2 (get it from https://github.com/gemfony/geneva)



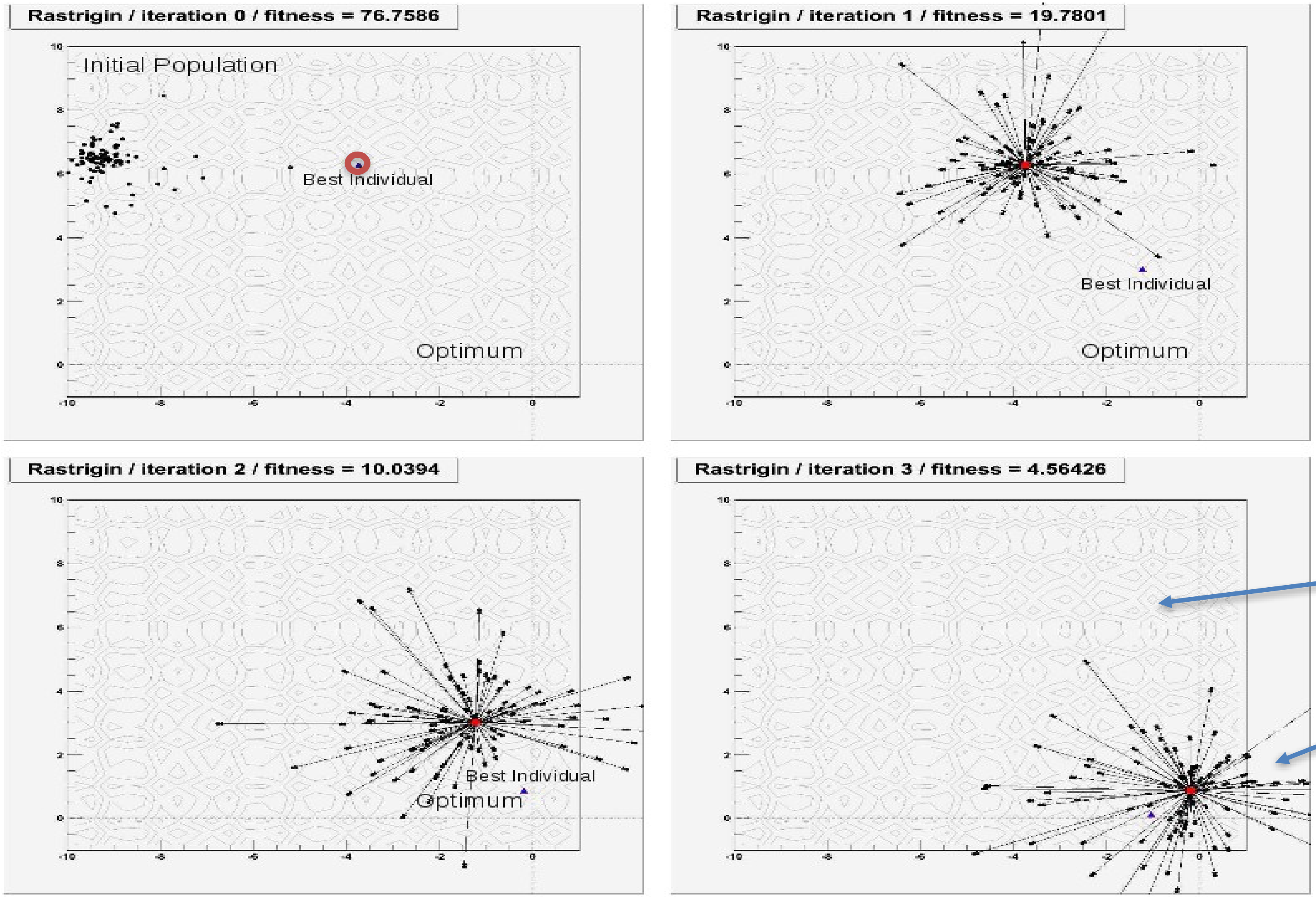
Our Vision

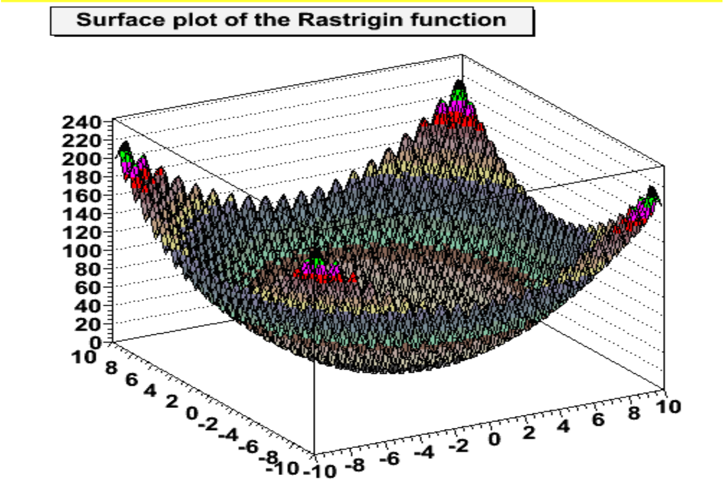
- To create a common Open Source optimisation framework, constantly developed and extended by physicists, computer scientists and software engineers according to professional standards, covering the most recent algorithms for massively parallel optimisation studies in research and industry
- To concentrate, activate and leverage fragmented knowledge in the field of parametric optimisation to enable each other solving even the most daunting optimisation problems



Picture credits: see last page

Dealing with Complex Quality Surfaces – Example: Evolutionary Algorithms





View in –z direction

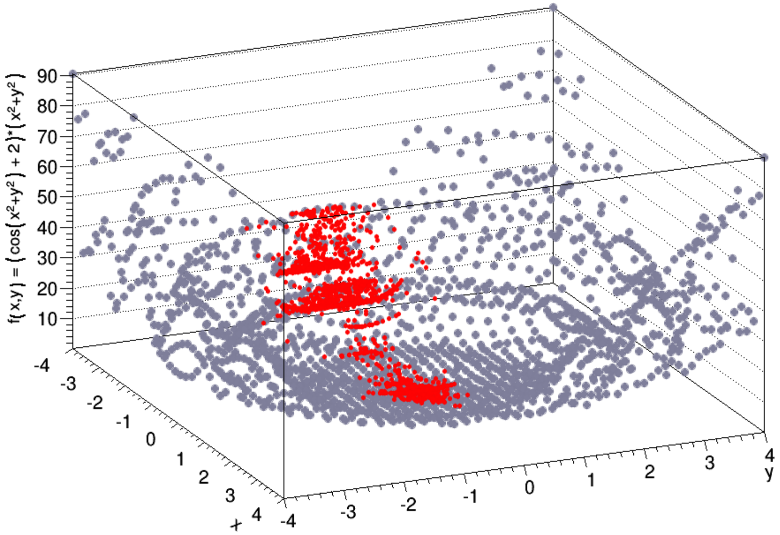
Parent-individual (red) and children

Source: Gemfony

Gemfony scientific

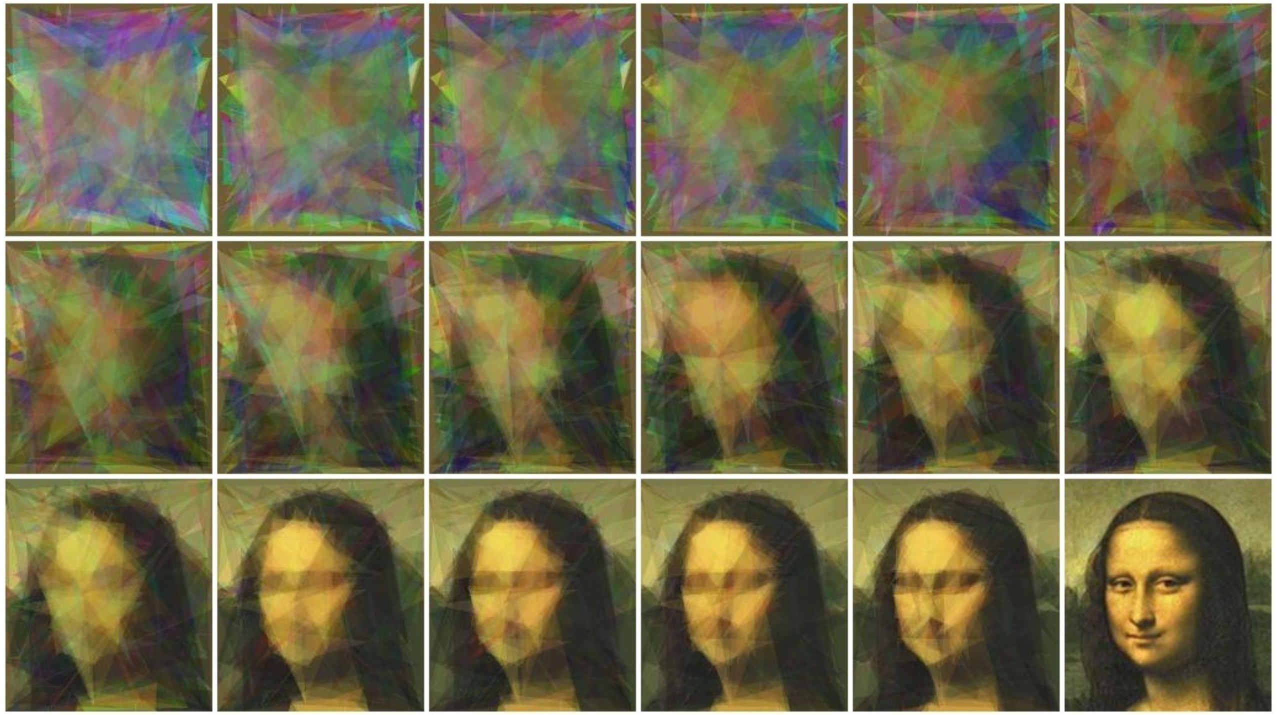
Dealing with high-dimensional parameter spaces

- Parameter scan / DOE only possible for low dimensions of the parameter space
  - Parameter scan with n evaluations per parameter in m dimensions: Need n^m evaluations
  - For 10 parameters and 10 evaluations each at 30 seconds: would require 9500 years CPU time …
- Thus: need dedicated optimisation algorithms that avoid visiting all of the parameter space
- As optimisation algorithms will typically call the solver hundreds or thousands of times, such optimisation problems will greatly benefit from parallelisation



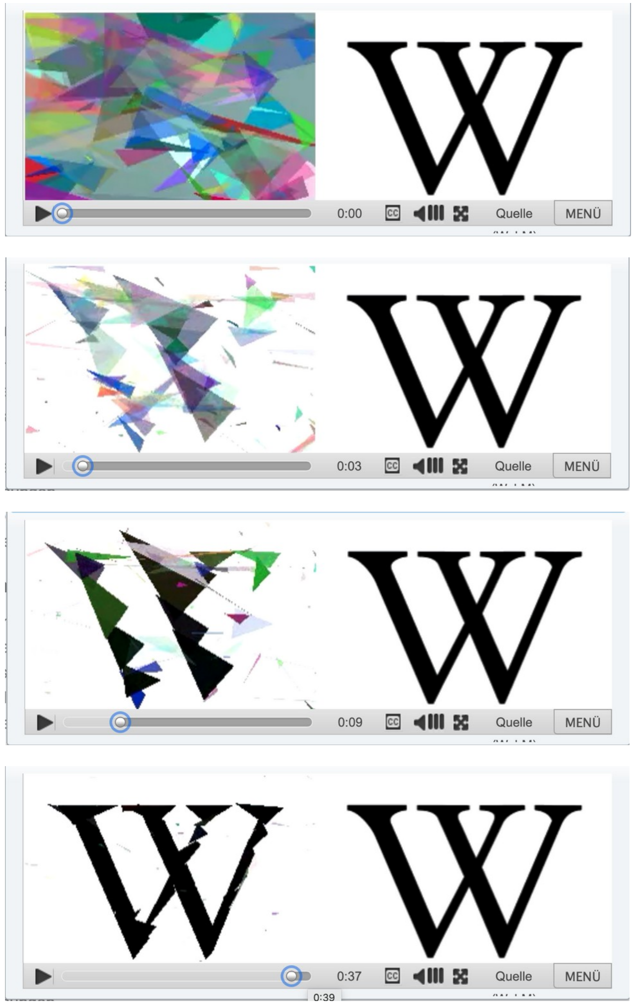
Source: Gemfony

Solving high-dimensional problems with EA



Source: Gemfony

3000 FP Parameters, constrained to [0,1]



Source: https://de.wikipedia.org/wiki/Geneva_(Software)

Asymptotic convergence:
Quick initial successes, followed by slower improvements
1500 parameters marks the limit of usefulness of this EA

Parallelisation: Comparatively Easy for Multi-Core Environments

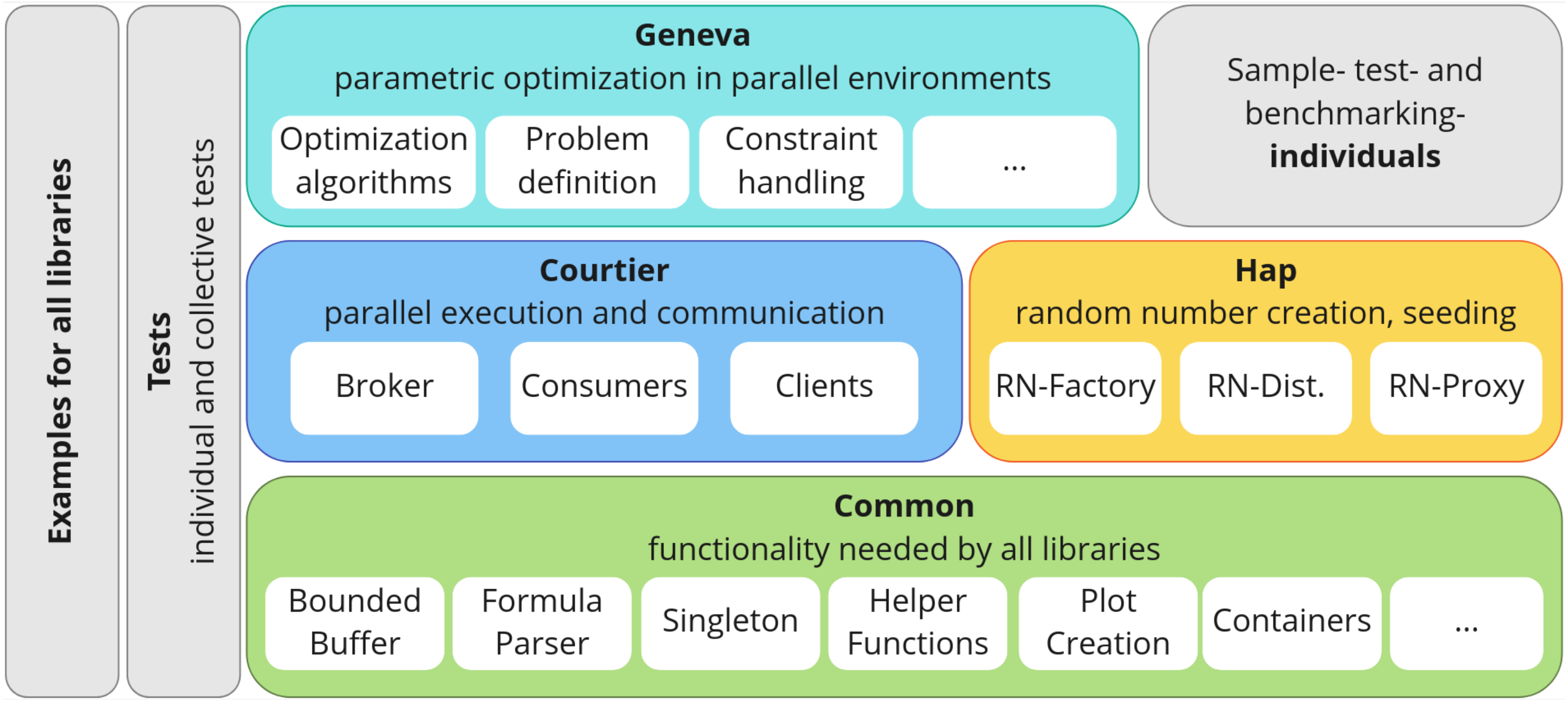
```
 Datei   Bearbeiten   Ansicht   Lesezeichen   Einstellungen   Hilfe
top - 16:03:19 up 118 days,  5:21,  2 users,  load average: 44.03, 30.19, 14.96
Tasks: 577 total,   1 running, 576 sleeping,   0 stopped,   0 zombie
Cpu(s): 99.9%us,  0.1%sy,  0.0%ni,  0.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Mem:  98998852k total, 13273640k used, 85725212k free,   366152k buffers
Swap: 101056504k total,    40224k used, 101016280k free, 11412904k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
25879 berlich   17   0 2684m 668m  28m S 4803.0  0.7  97:24.42 GOmegaPiApp
25868 berlich   15   0 13140 1492  820 R  5.8  0.0   0:02.31 top
25603 berlich   15   0 90128 1824 1088 S  0.0  0.0   0:00.08 sshd
25604 berlich   15   0 66060 1620 1252 S  0.0  0.0   0:00.05 bash
25653 berlich   15   0 90128 1828 1088 S  0.0  0.0   0:00.06 sshd
25654 berlich   15   0 66060 1596 1240 S  0.0  0.0   0:00.01 bash
```

- Even large systems may be saturated by suitable work-loads (long evaluation)
- Example uses Evolutionary Strategies, as implemented in Geneva
- For large populations, resembles an “embarrassingly parallel“ problem

Source: Gemfony

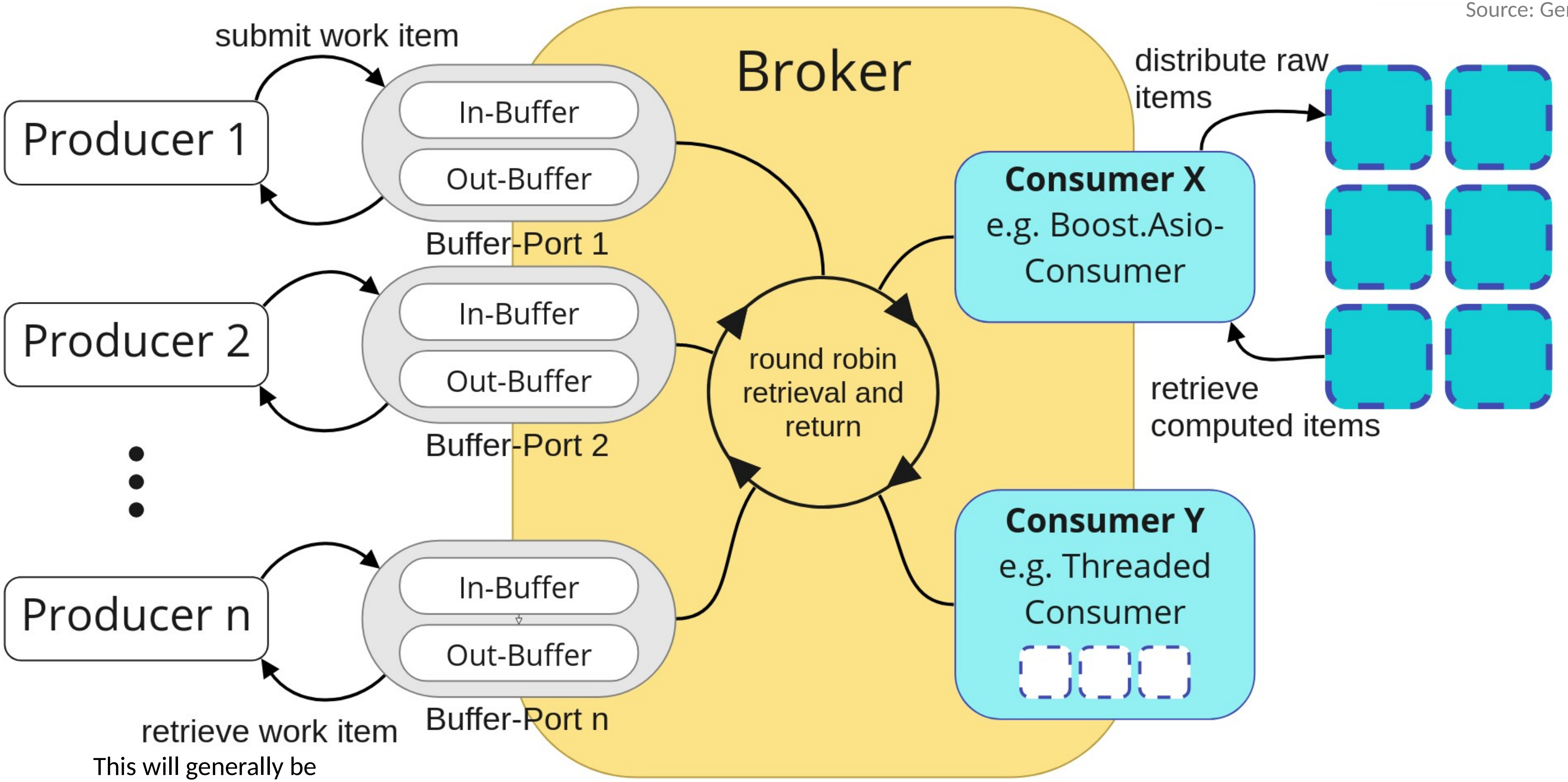
System Architecture of Geneva



Common sublibrary: central functionality as thread pool, logger, parser, …
Geneva sublibrary: functionality specific for parametric optimisation
Courtier sublibrary: template library for parallelisation
Hap sublibrary: centralised generation of randomised numbers
Automated unit tests, benchmarks and examples

Parallelisation: More Complex in Distributed Environments

Source: Gemfony



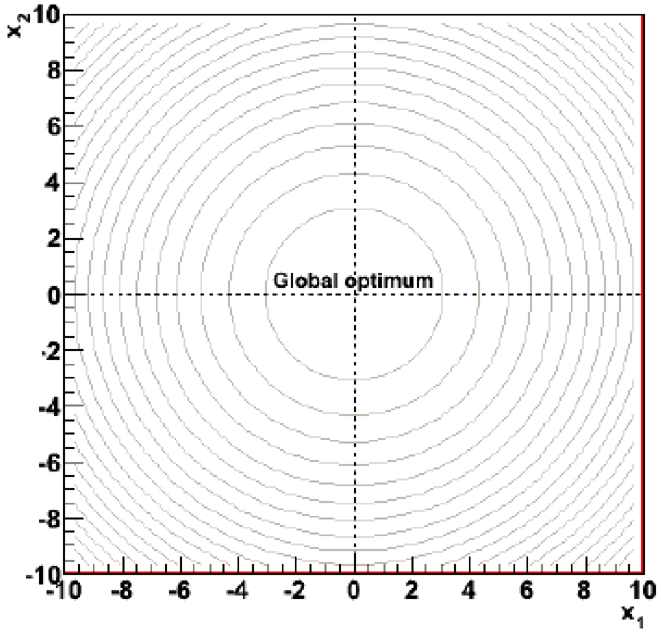
This will generally be
An optimisation algorithm

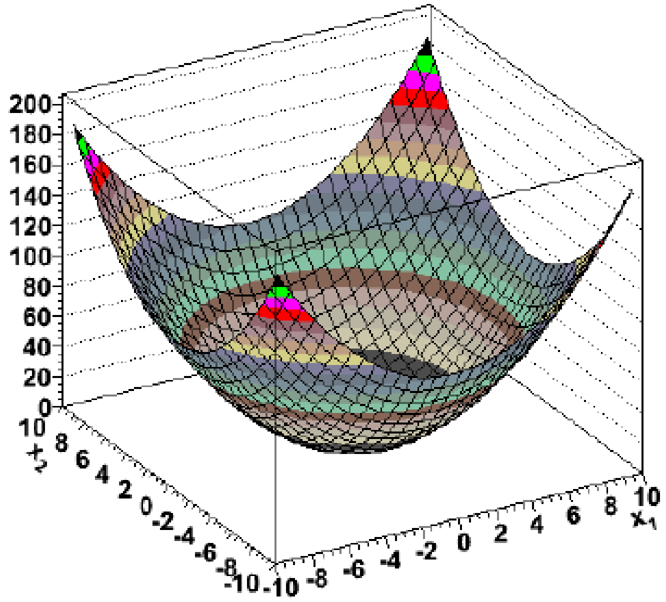
Using Geneva

Manual see https://www.gemfony.eu/fileadmin/documentation/geneva-manual.pdf

- Defining a first optimisation problem
- In an n-dimensional paraboloid, the „quality“ of the parameter set (n floating point numbers in this case) is defined as follows:

$$f(x_1, x_2, \dots, x_n) = \sum_{i=1}^{n} x_i^2 = x_1^2 + x_2^2 + \dots + x_n^2$$





Class definition of a 2 dimensional parabola

- The following class lets us search for the minimum of a two-dimensional parabola
- It is derived from GParameterSet, the base class of all individuals

Class GParaboloidIndividual2D

```
class GParaboloidIndividual2D :public GParameterSet
{
public:
  GParaboloidIndividual2D(); // default constructor
  GParaboloidIndividual2D(const GParaboloidIndividual2D&); // copy constructor
  virtual ~GParaboloidIndividual2D(); // destructor

protected:
  // Loads the data of another GParaboloidIndividual2D
  virtual void load_(const GObject*);
  // Creates a deep clone of this object
  virtual GObject* clone_() const;

  // Calculates the object's quality
  virtual double fitnessCalculation();

private:
  // Make the class accessible to Boost.Serialization
  friend class boost::serialization::access;

  // Triggers serialization of this class and its base classes.
  template<typename Archive>
  void serialize(Archive & ar, const unsigned int) {
    using boost::serialization::make_nvp;
    // Serialize the base class
    ar & BOOST_SERIALIZATION_BASE_OBJECT_NVP(GParameterSet);
    // Add other variables here like this:
    // ar & BOOST_SERIALIZATION_NVP(sampleVariable);
  }

  const double PAR_MIN_; // Lower boundary for parameters
  const double PAR_MAX_; // Upper boundary for parameters
};
```

The constructor – adding parameters

Listing 11.2: The constructor of the GParaboloidIndividual2D class

```
GParaboloidIndividual2D::GParaboloidIndividual2D()
        : GParameterSet()
        , PAR_MIN_(-10.)
        , PAR_MAX_(10.)
{
        for(std::size_t npar=0; npar<2; npar++) {
                // GConstrainedDoubleObject is constrained to [PAR_MIN_:PAR_MAX_[
                boost::shared_ptr<GConstrainedDoubleObject>
                   gcdo_ptr(new GConstrainedDoubleObject(PAR_MIN_, PAR_MAX_));
                // Add the parameters to this individual
                this->push_back(gcdo_ptr);
        }
}
```

The fitness calculation

```
double GParaboloidIndividual2D::fitnessCalculation(){
  double result = 0.; // Will hold the result
  std::vector<double> parVec; // Will hold the parameters

  this->streamline(parVec); // Retrieve the parameters

  // Do the actual calculation
  for(std::size_t i=0; i<parVec.size(); i++) {
          result += parVec[i]*parVec[i];
  }

  return result;
}
```

The main function

```
using namespace Gem::Geneva;
int main(int argc, char **argv) {
        Go2 go(argc, argv, "config/go2.json");

        //----------------------------------------------------------------------
        // Initialize a client, if requested
        if(go.clientMode()) return go.clientRun();

        //----------------------------------------------------------------------
        // Add individuals and algorithms and perform the actual optimization cycle

        // Make an individual known to the optimizer
        boost::shared_ptr<GParaboloidIndividual2D> p(new GParaboloidIndividual2D());
        go.push_back(p);

        // You could add an algorithm to the Go2 class here, which would always be
        // executed first. Not specifiying any algorithms results in the default
        // default algorithm, unless other algorithms specified on the command line.
```

Using JSON for the configuration
Here: GEvolutionaryAlgorithm.json

```
    }
},
"maxIteration": {
    "comment": "The maximum allowed number of iterations",
    "default": "1000",
    "value": "1000"
},
"minIteration": {
  "comment": "The minimum allowed number of iterations",
  "default": "0",
  "value": "0"
},
"maxStallIteration": {
    "comment": "The maximum allowed number of iterations without improvement",
    "comment": "0 means: no constraint.",
    "default": "20",
    "value": "20"
},
"indivdualUpdateStallCounterThreshold": {
    "comment": "The number of iterations without improvement after which",
    "comment": "individuals are asked to update their internal data structures",
    "comment": "through the actOnStalls() function. A value of 0 disables this check",
    "default": "0",
    "value": "0"
},
"reportIteration": {
    "comment": "The number of iterations after which a report should be issued",
    "default": "1",
    "value": "1"
},
```

With auto-generated command line options

```
rberlich@Ubuntu-1910-eoan-64-minimal:~/Progs/geneva-build/examples/geneva/02_GParaboloid2D$ ./GParaboloid2D --help
Usage: ./GParaboloid2D [options]:

Basic options:
  -h [ --help ]                        Emit help message
  --showAll                            Show all available options
  -a [ --optimizationAlgorithms ] arg  A comma-separated list of optimization
                                       algorithms, e.g. "arg1,arg2". 5
                                       algorithms have been registered:
                                       ea:  Evolutionary Algorithm
                                       gd:  Gradient Descent
                                       ps:  Parameter Scan
                                       sa:  Simulated Annealing
                                       swarm:  Swarm Algorithm

  -f [ --cp_file ] arg (=empty)        A file (including its path) holding a
                                       checkpoint for a given optimization
                                       algorithm
  --client                             Indicates that this program should run
                                       as a client or in server mode. Note that
                                       this setting will trigger an error
                                       unless called in conjunction with a
                                       consumer capable of dealing with clients
  --maxClientDuration arg (=00:00:00)  The maximum runtime for a client in the
                                       form "hh:mm:ss". Note that a client may
                                       run longer as this time-frame if its
                                       work load still runs. The default value
                                       "00:00:00" means: "no time limit"
  -c [ --consumer ] arg (=stc)         The name of a consumer for brokered
                                       execution (an error will be flagged if
                                       called with any other execution mode
                                       than (2) ). 4 consumers have been
                                       registered:
                                       asio:  GAsioConsumerT
                                       beast:  GWebsocketConsumerT
                                       sc:  GSerialConsumerT
```

First output

```
Seeding has started
Starting an optimization run with algorithm "Evolutionary Algorithm"
0: 64.6073443050163
1: 25.9597623490252
2: 8.89715425355864
3: 1.45564799125829
4: 0.861887897798893
[...]
999: 7.37074272148514e-13
End of optimization reached in algorithm "Evolutionary Algorithm"
Done ...
```

In the Client Server mode many clients/individuals can run in parallel and contribute to solving a complex problem

Geneva Application in Hadron Theory

Hadronic reaction amplitudes for FAIR

- A framework for predicting and analysing final-state interactions for the FAIR experiments is beeing developed
- This requires massively parallel computing, up to 50 and more coupled-channels needed
- Reaction amplitudes are derived from effective Lagrangians where coupled-channel unitarity and the implications of micro-causality (dispersion relations) have been implemented (isobar models are not good enough)
- A subset of the parameters can be derived from the quark-mass dependence of existing QCD lattice data and/or fits to existing data
- Conventional fitting routines like Minuit are not suitable for such highly non-linear problems – chisquare surface is too rough (gradients are expensive and not stable)
- In order to avoid local minima and to be able to find the best possible solution an Evolutionary Algorithm with reasonably high population is being used

Projects done with Geneva

On finite volume effects in the chiral extrapolation of baryon masses
M.F.M. Lutz, R. Bavontaweepanya, C. Kobdaj, K. Schwarz
Published in Physical Review D 01/2014 90(5)

From Hadrons at Unphysical Quark Masses to Coupled-Channel Reaction Dynamics in the Laboratory
M.F.M. Lutz, Xiao-Yu Guo, Y. Heo
Published in: JPS Conf. Proc. 26 (2019) 022022

On a first order transition in QCD with up, down, and strange quarks
Xiao-Yu Guo, Y. Heo, M.F.M. Lutz
Published in: Eur. Phys. J. C 80 (2020) 3, 260

A generalised Higgs potential with two degenerate minima for a dark QCD matter scenario
M.F.M. Lutz, Y. Heo, Xiao-Yu Guo
Published in: Eur. Phys. J. C 80 (2020) 4, 322

Low-energy constants from charmed baryons on QCD lattices
Y. Heo, Xiao-Yu Guo, M.F.M. Lutz
Published in: Phys. Rev. D 101 (2020) 5, 054506

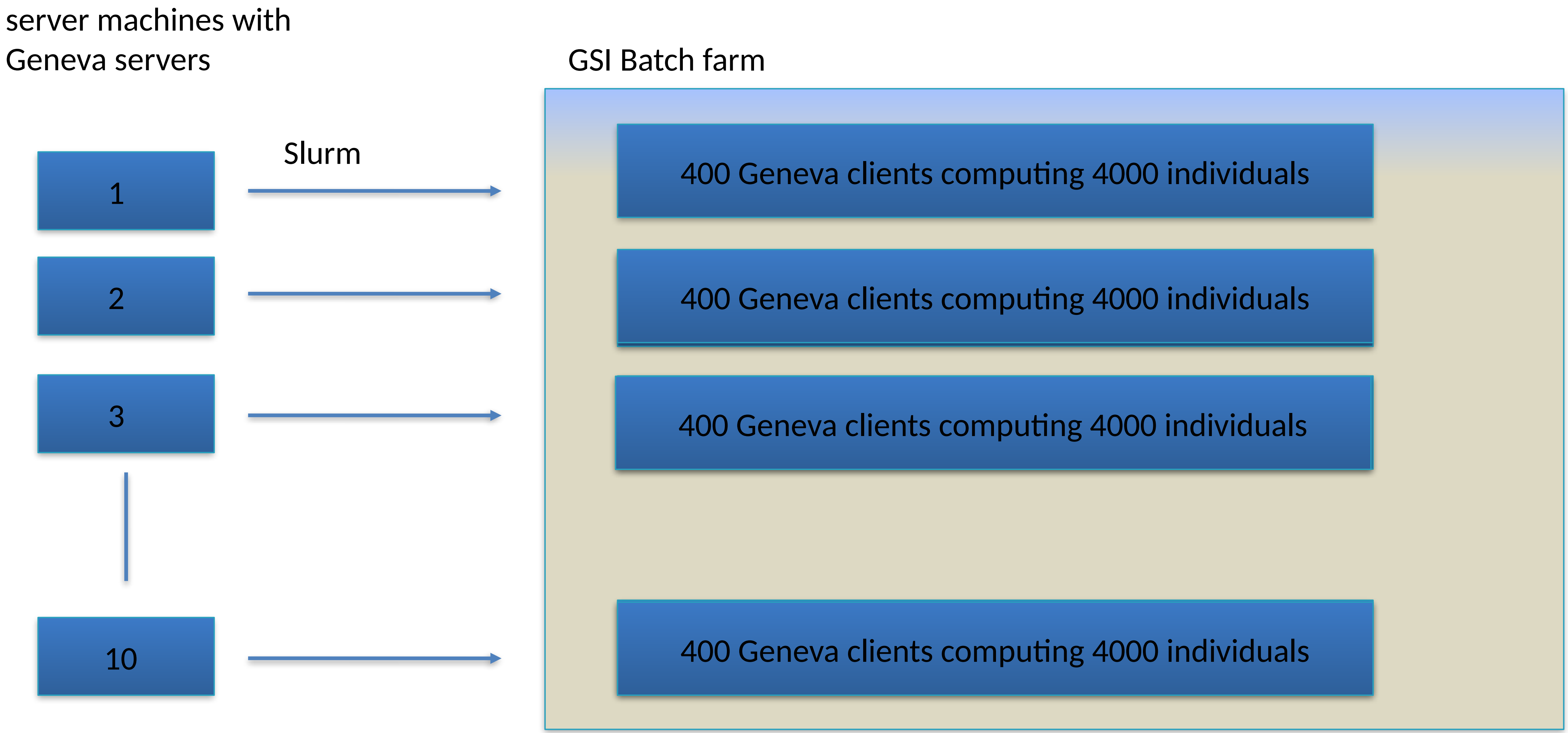
On the axial-vector form factor of the nucleon and chiral symmetry
Matthias F.M. Lutz, Ulrich Sauerwein, Rob G.E. Timmermans
Published in: Eur.Phys.J.C 80 (2020) 9, 844

Coupled-channel dynamics with chiral long-range forces in the open-charm sector of QCD
Matthias F.M. Lutz, Xiao-Yu Guo, Yonggoo Heo, C.L. Korpa
Published in: Phys.Rev.D 106 (2022) 11, 114038

Low-energy constants in the chiral Lagrangian with baryon octet and decuplet fields from Lattice QCD data on CLS ensembles
Matthias F.M. Lutz, Yonggoo Heo, Xiao-Yu Guo
Published in Eur.Phys.J.C 83 (2023) 5, 440

Coupled-channel system with anomalous thresholds and unitarity
Csaba L. Korpa(, Matthias F.M. Lutz, Xiao-Yu Guo, Yonggoo Heo
Published in: Phys.Rev.D 107 (2023) 3, L031505

Geneva Cluster @ GSI
example case: 10 minutes compute time for one solution, 10 x 400 clients, 10 x 4000 population, 1000 iterations, one week of compute time in total



Contributions to Geneva in context of Hadron Theory

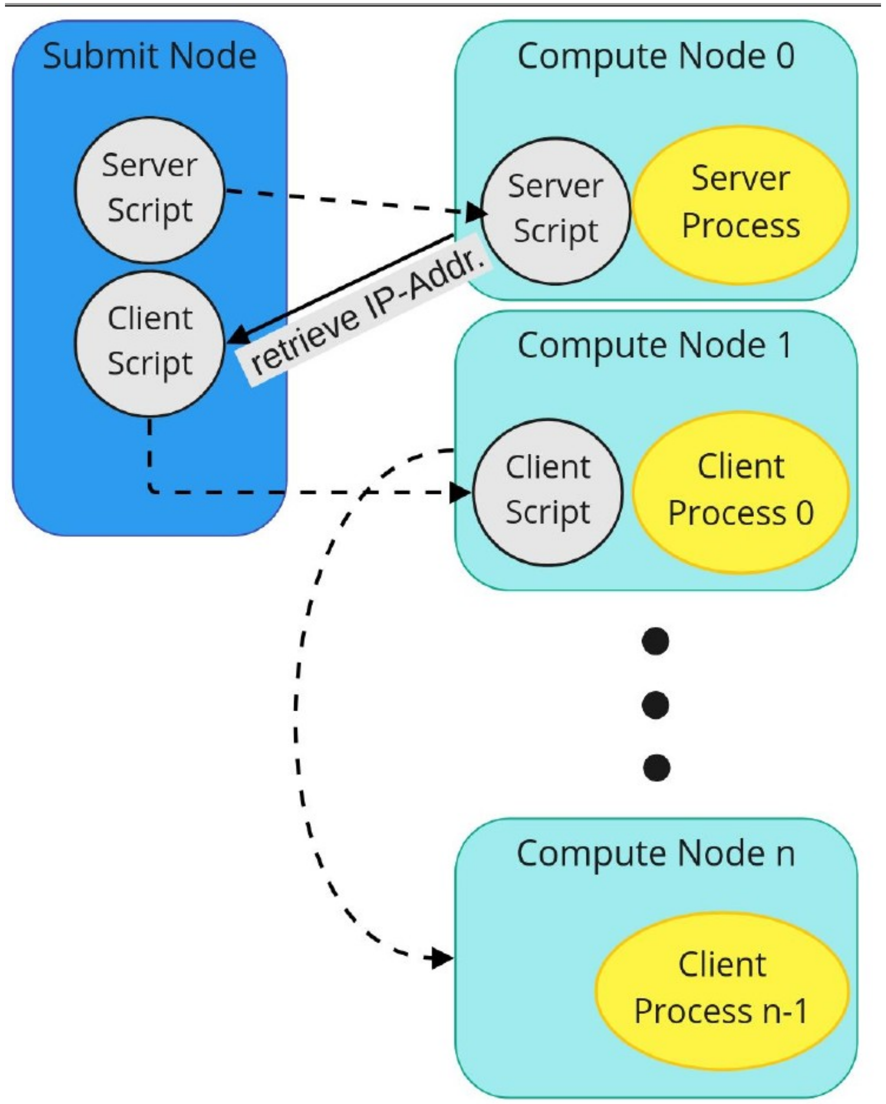
- Geneva client-server communication
  - transition from Boost.ASIO-Sockets to Beast Websockets
  - heartbeat option allows better client control
  - less load on server, higher number of clients, higher efficiency
- Checkpoints
  - iterations are stored in checkpoints (text, xml or binary)
  - iterations can be continued later by loading the checkpoint file
- **MPI Consumer** for running optimisations on HPC clusters

Boost.Asio is a cross-platform C++ library for network and low-level I/O programming that provides developers with a consistent asynchronous model using a modern C++ approach.

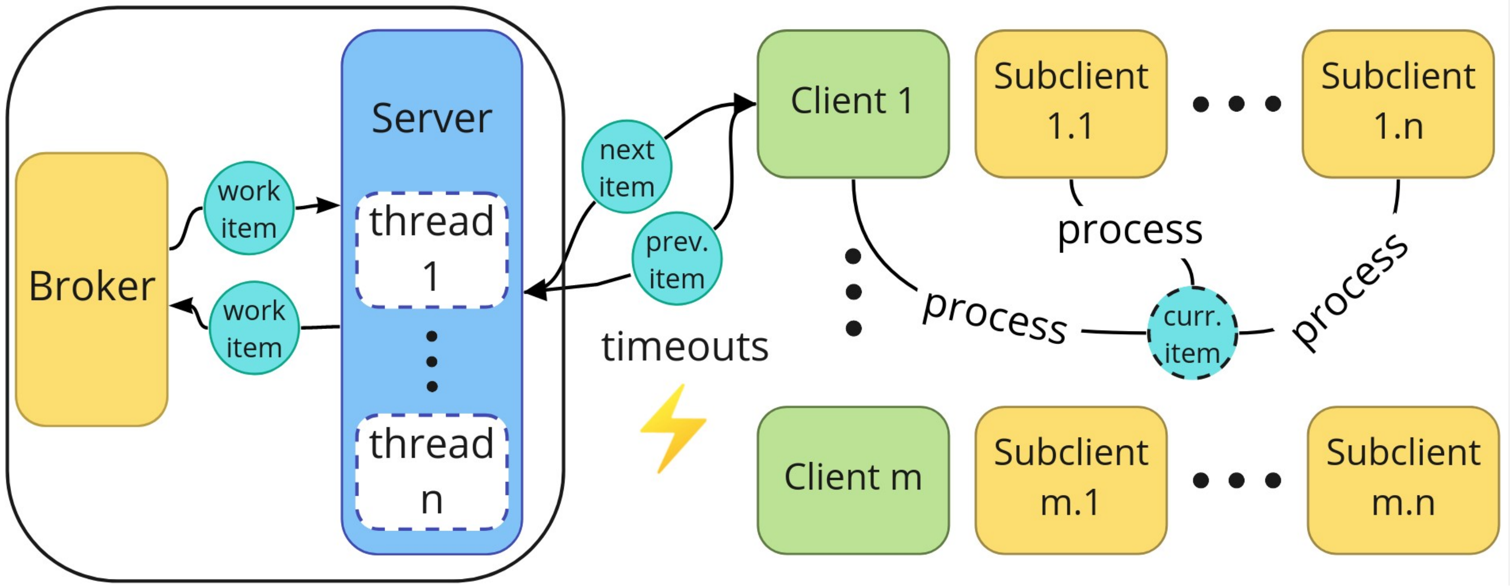
MPI Consumer for Geneva

motivation

- Geneva has not been optimised for HPC computing
- Geneva had no integration with common cluster scheduling systems
- Geneva did not allow for distributed parallelisation of user defined cost function
  - If requirements go beyond the resources of a single node
- Improved user experience by doing automatic configuration on cluster

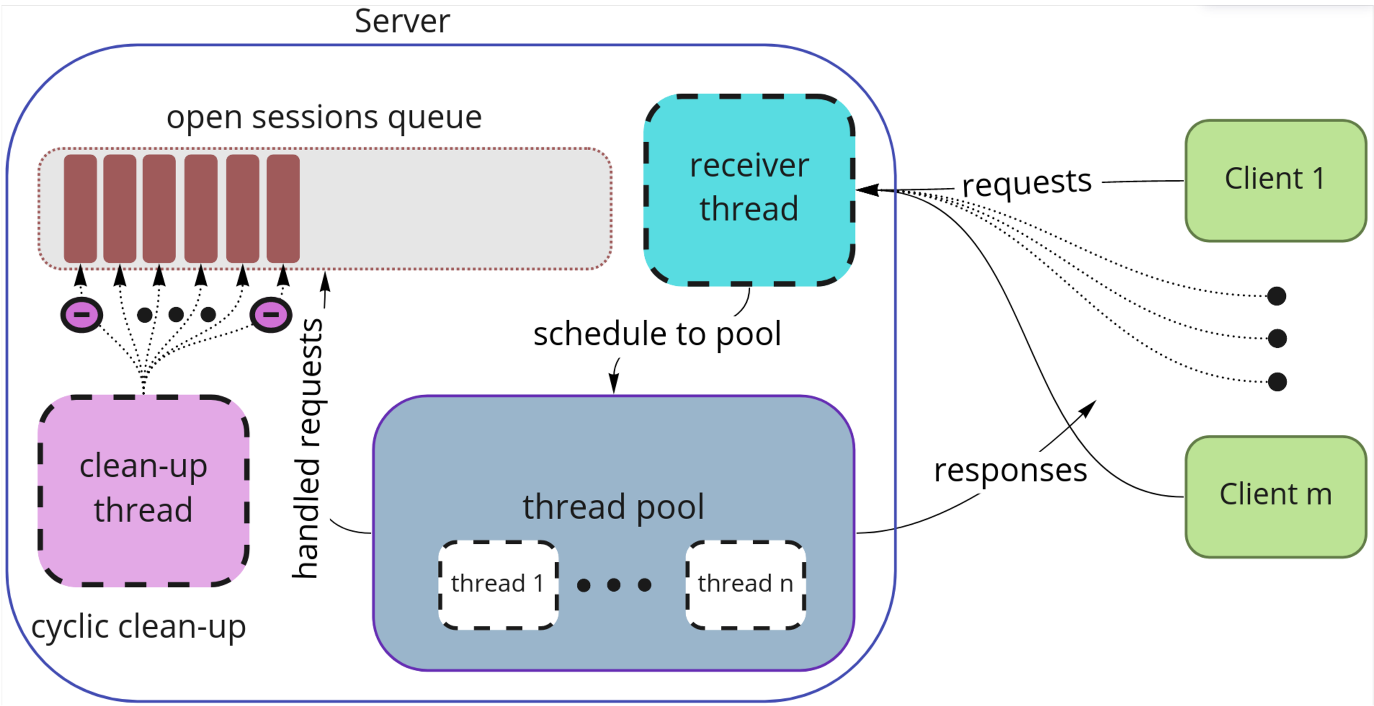


MPI Consumer: System design overview



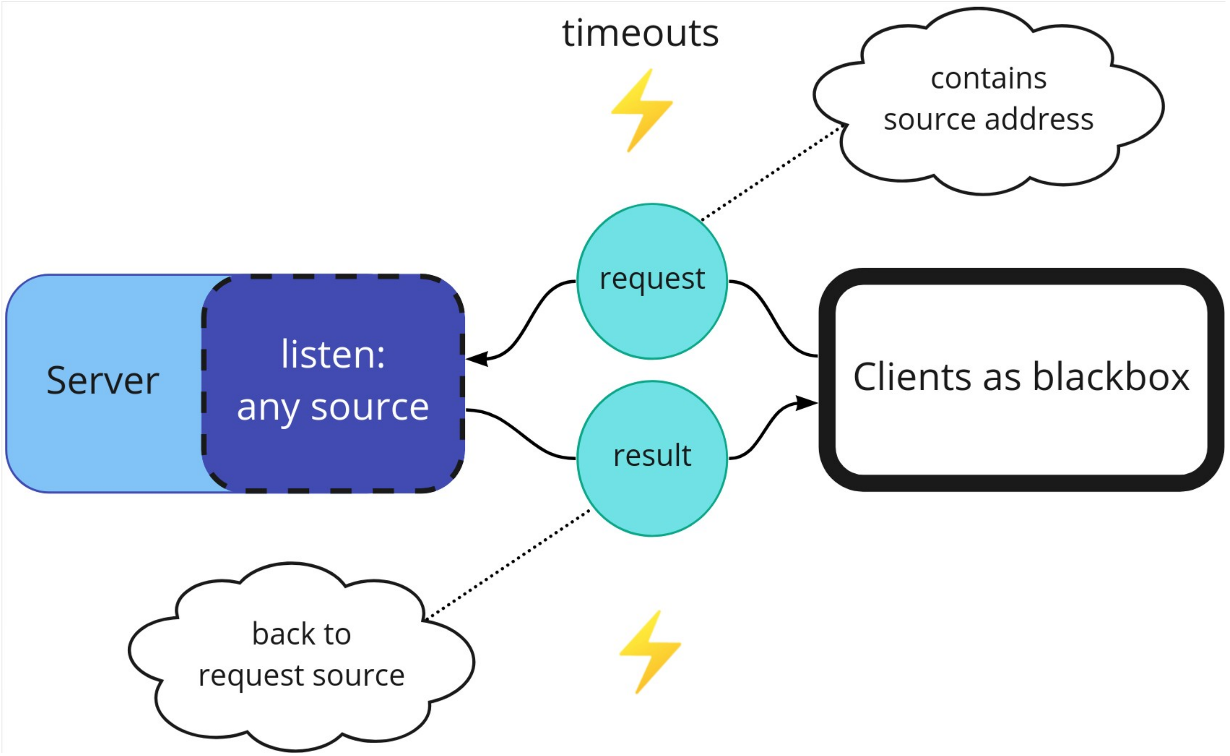
- Server waits for clients to request work items
- If requested, the work item is taken from the broker and sent to the requesting client
- Using asynchronous requests, clients may send next request while processing the work item received
- For parallelising the quality function itself via additional class evaluation function can be distributed to multiple machines (sub clients)

MPI Consumer: Multithreading



- Receiver thread permanently waits for requests from clients
- Execution streams at thread pool deal with processing received requests
- Open sessions queue keeps handled requests whose asynchronous sending to client has not yet been performed
- Clean-up-thread checks for completed, timeouted or erroneous tasks

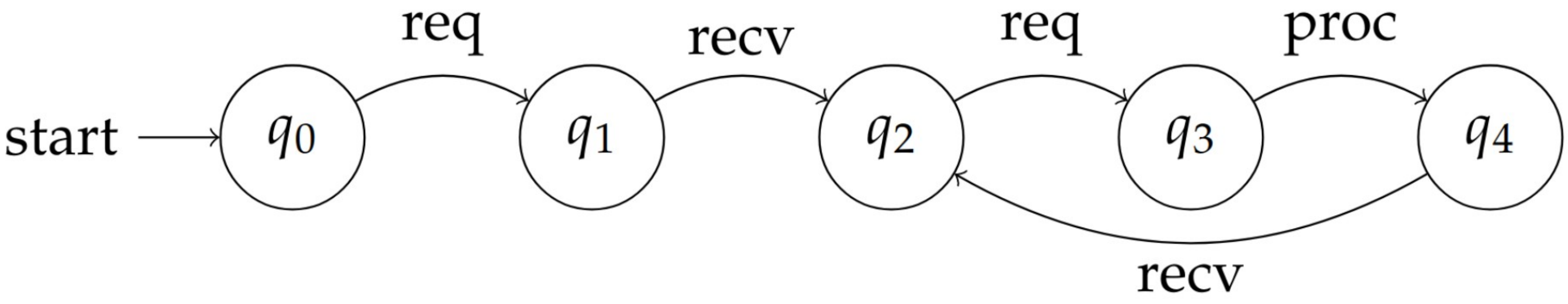
MPI Consumer: Fault tolerance



Asynchronous network operation is initiated

- Repeated checks if operation completed or run into timeout
- If time frame exceeded connection is terminated, resources are released, clients are shut down in controlled manner

MPI Consumer: Asynchronous requests

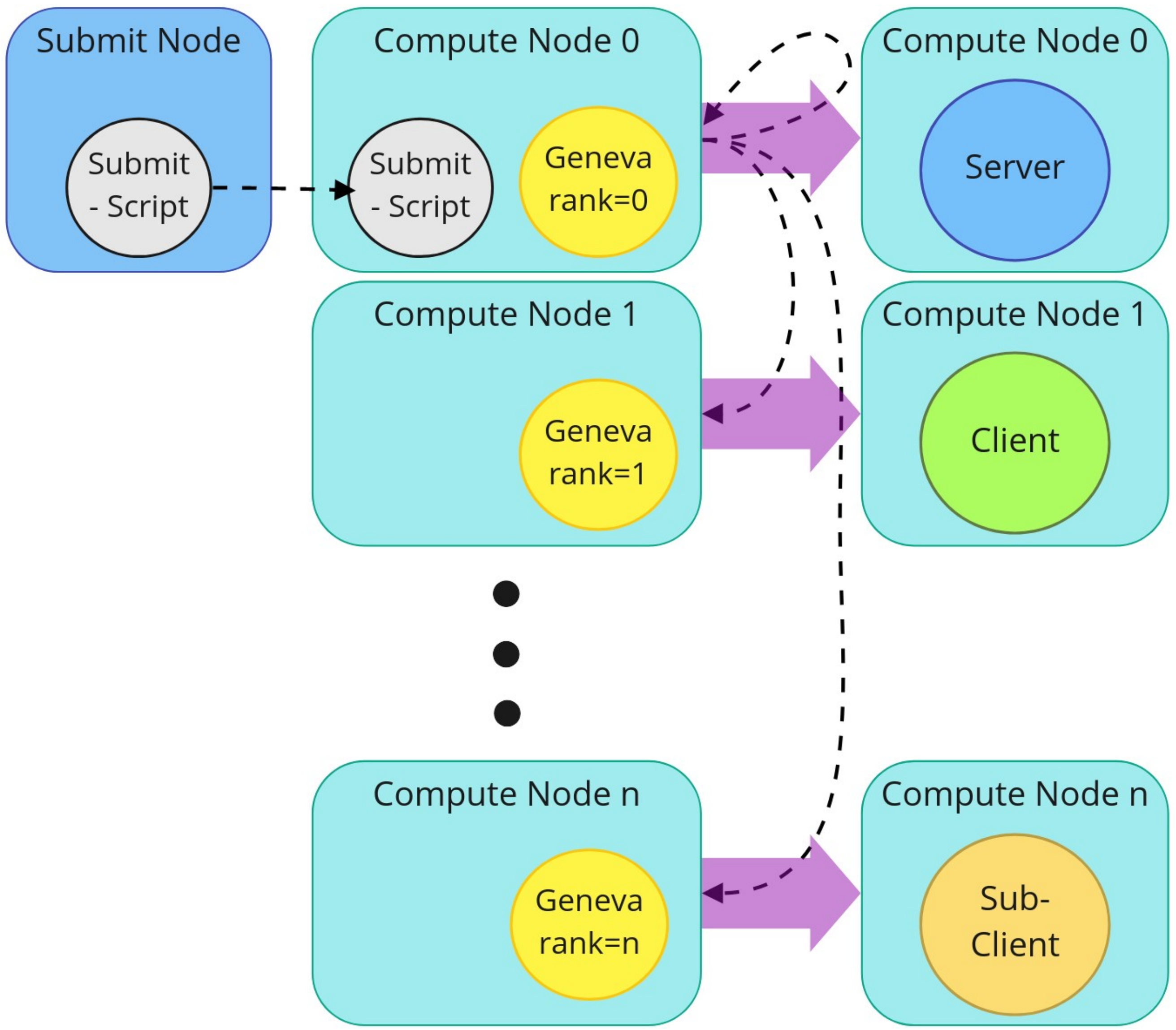


Without asynchronous requests speedup is limited by Amdahl's law

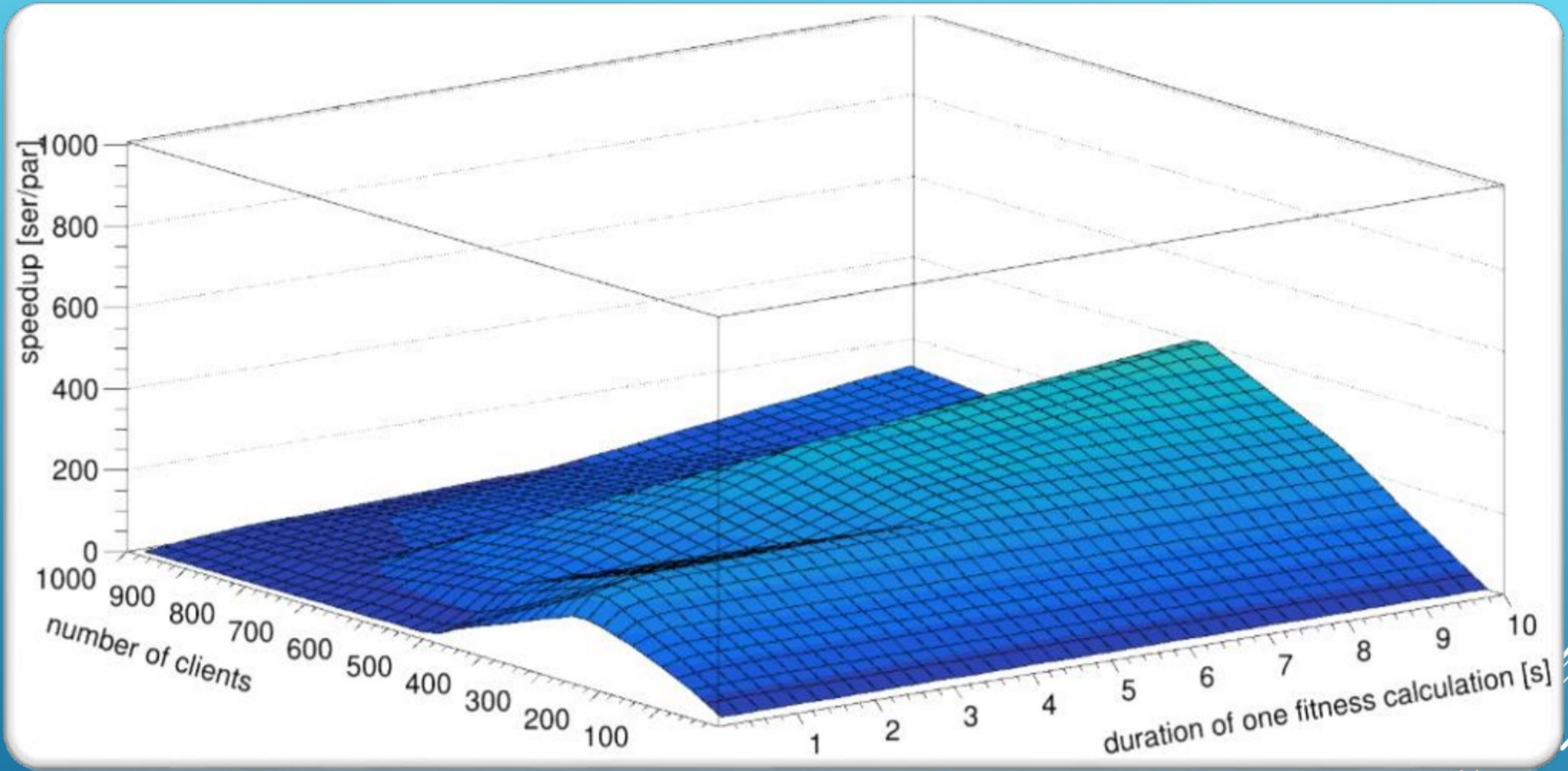
- Clients send request to server and fill local queues
- In Q2 clients are in cycle
  - New work item is requested, available work items are computed
  - Waiting times for server responses are filled with meaningful computations

MPI Consumer: Automatic configuration

- Scheduling system (e.g. Slurm) allocates certain number of compute nodes
- Configured number of Geneva instances are started on allocated compute nodes
- The processes configure their role at run time using their MPI rank (environment variable by scheduling system)
- Boost.Asio and Boost.Beast consumers require server's IP address and port at client startup time. Therefore 2 step process was required: first schedule server then schedule clients



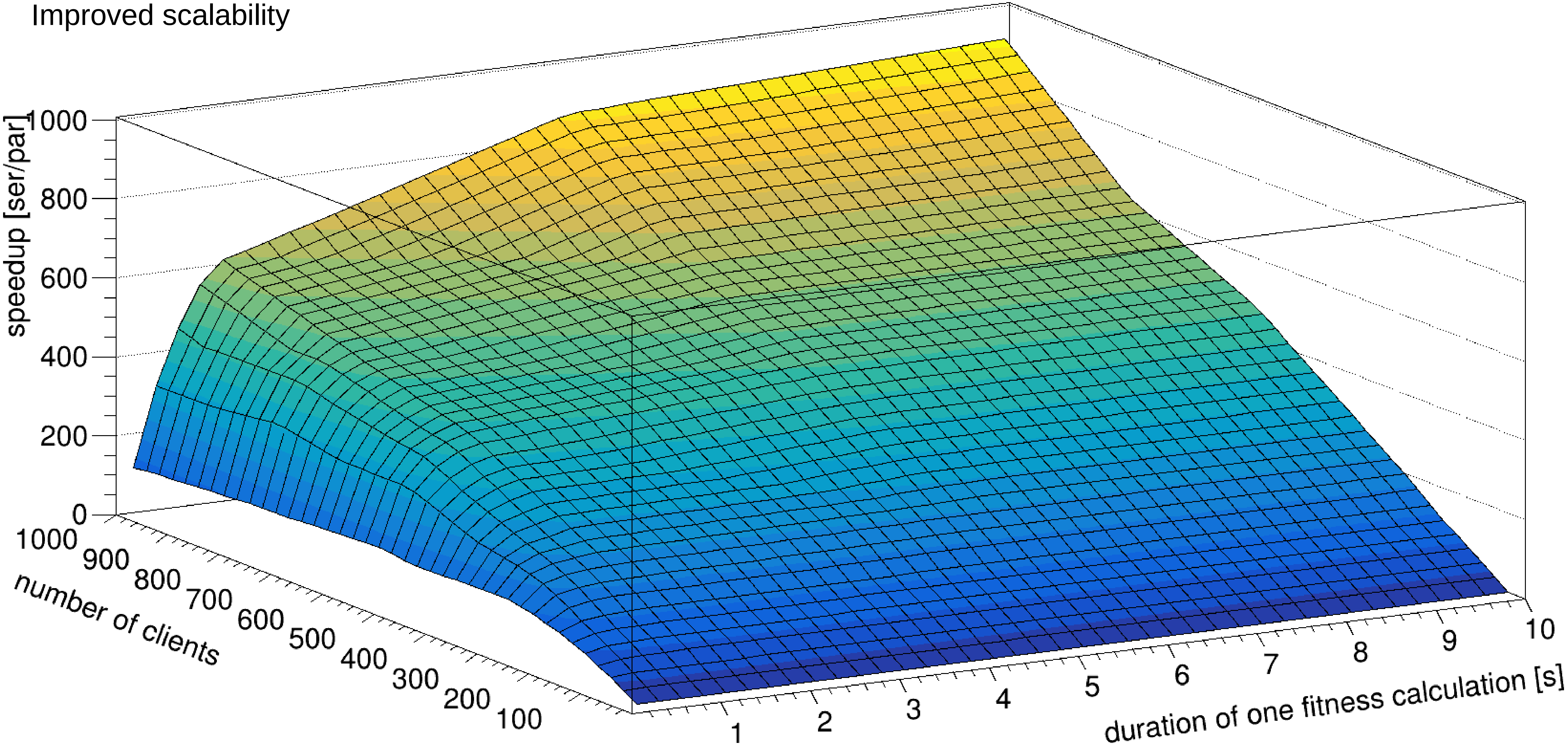
Benchmarking

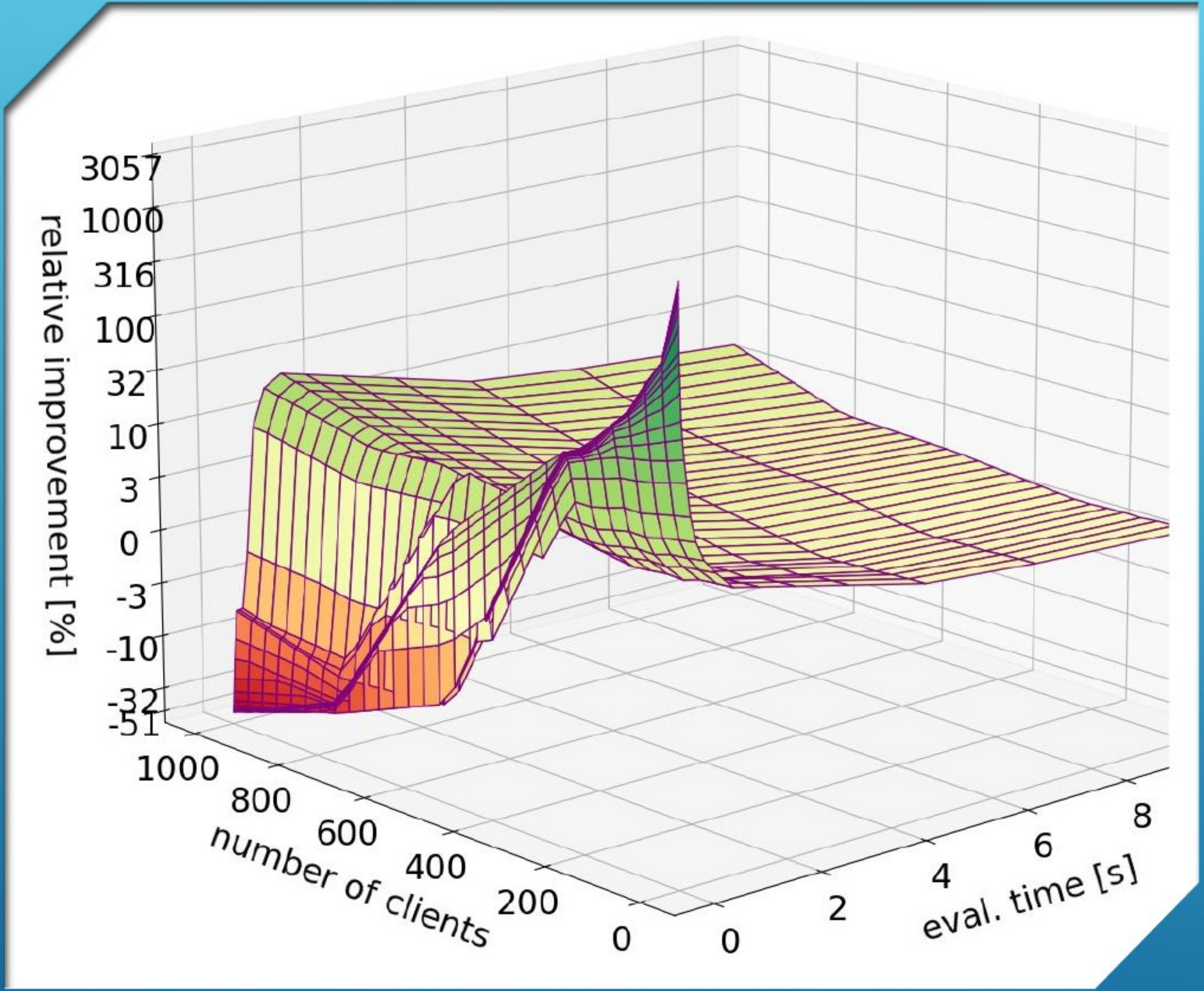


BOOST.ASIO CONSUMER

- LIMITED SCALABILITY
- TYPICALLY AT GSI: 5+ SEC EVALUATION TIME

Boost.Beast Consumer with 64 Threads
Improved scalability





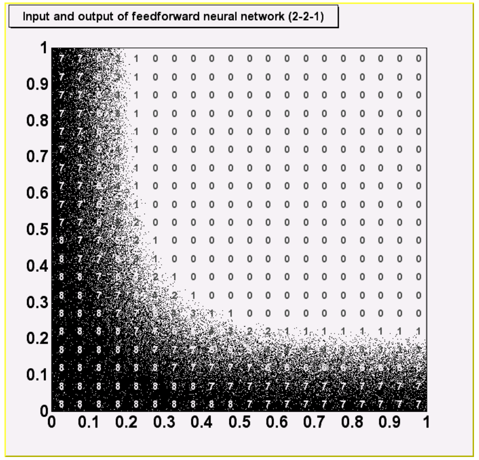
MPI VS. BEAST

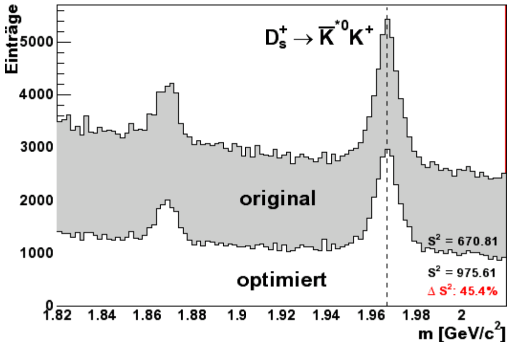
- IMPROVEMENTS FOR ALL RELEVANT WORKLOADS

Lessons learned

Some History

- Started as a means of training neural networks 1994 (Ruhr-Universität Bochum, Crystal Barrel)
- Extended to optimise particle physics analysis (Ruhr-Universität Bochum, BaBar)
- Rewrite at Karlsruhe Institute of Technology and subject of a Spin-Off
  - Hence drifted off into consulting, software engineering …
- Usage at GSI and in Industry
- Geneva has grown over time, from 1000 LOC to about 130000 LOC
- Need to build community
- **The approach has proven to be successful and we would appreciate your help in a more dynamic user community and to increased development activity**





Source: Gemfony

C++ as an implementation project for distributed systems

- C++ wants to be the ideal language
  - The “academically perfect“ language
  - High-speed, close to the bare metal
  - All possibilities reserved (but easy to shoot yourself in the foot)
  - „Design by committees“ (plural …)
- Minimal focus on standard libraries, infrastructure, surrounding
  - No networking in the standard after 40 years ?!??
  - Many missing multithreading constructs (think „threadpool“)
- Language would have long been dead, except
  - There is a large pool of high quality libraries out there (think „Boost“)
  - Highly knowledgable community
  - If you do have the tools and the knowledge, using C++ can be a joy! ^^
- C++ is hence still in wide use throughout science and industry

C++ as an implementation project for distributed systems

- Lack of networking constructs has meant for Geneva: build your own
  - Initially based on MPI, then Boost.ASIO, then Boost.Beast (Websockets)
- Majority of work went into this
  - This was NOT the intention!
  - Not the core business of Geneva (but arguably what makes it useful)
  - Many hard to track bugs, and difficult to find suitable test environments (I am paying > 1000 Euros per year for root servers)
- A useful structure has evolved from this
  - A random number factory
  - Brokerage
  - Test-infrastructure (in-class definition, decentral execution)
- Still: could be done (much) better
  - But lets not try to do the same mistake C++ did …
  - The focus in the future should be much more on optimisation algorithms

Conclusion and Outlook

Conclusion and Outlook

- Geneva is an efficient client/server tool for doing distributed optimisation within an HPC environment
  - mainly using Evolutionary Algorithms
  - Used with up to 400 clients with populations up to 4000
  - Available under the Apache License v2
- May need refactoring and needs a larger community
  - Needs ideas both for optimisation and for the clustering part
- Future work
  - adding more reliable optimisation algorithms
  - Adding additional Consumers
  - increasing scalability (although already close to optimum)
  - starting inter-site optimisation on Grids/Clouds
  - Geneva Spack package and Geneva Singularity Container for easy use
  - Simplify Geneva interface even more for common usage patterns

Thank you

Do contact us in case of questions:
kilian.schwarz@desy.de
r.berlich@gemfony.eu
m.lutz@gsi.de
j.wessner@gsi.de

If you want to try Geneva:
https://github.com/gemfony/geneva
http://www.gemfony.eu

More information see published article about Geneva:

Parametric Optimization on HPC Clusters with Geneva

Jonas Weßner, Rüdiger Berlich, Kilian Schwarz, Matthias F.M. Lutz

Published in: Comput.Softw.Big Sci. 7 (2023) 1, 4

Masthead Gemfony

Address	Gemfony scientific UG (haftungsbeschränkt) Hauptstraße 2 76344 Eggenstein-Leopoldshafen - Germany
Telefone	+49(0)7247/934278-0
Fax	+49 (0)7247 934 2781
Email	
Registered at	Amtsgericht Mannheim (Germany)
Registration-Id	HRB 710566
Ust.-Id	DE274421406
Managing Director	Dr. Rüdiger Berlich

Material used

Slide	Figure	Source
The Geneva library collection	Car in puzzle	Image courtesy of Simon Howden at FreeDigitalPhotos.net
The Geneva library collection	Wind turbines in puzzle	http://www.flickr.com/photos/pebondestad/3533177131/sizes/l/in/photostream/Creative Commons Attribution 2.0; By Pål Espen Bondestad.
The Geneva library collection	Particle decay	https://en.wikipedia.org/wiki/File:CMS_Higgs-event.jpg ; Creative Commons Attribution Share-Alike 3.0; By CERN
Other slides	Other pictures	Gemfony scientific + GSI

Thank you very much

Kontakt

Deutsches Elektronen-Synchrotron DESY

www.desy.de

Kilian Schwarz
IT/Scientific Computing
kilian.schwarz@desy.de
040 8998 (9) 2596