Sussex HPC workshop, 14-12-2011

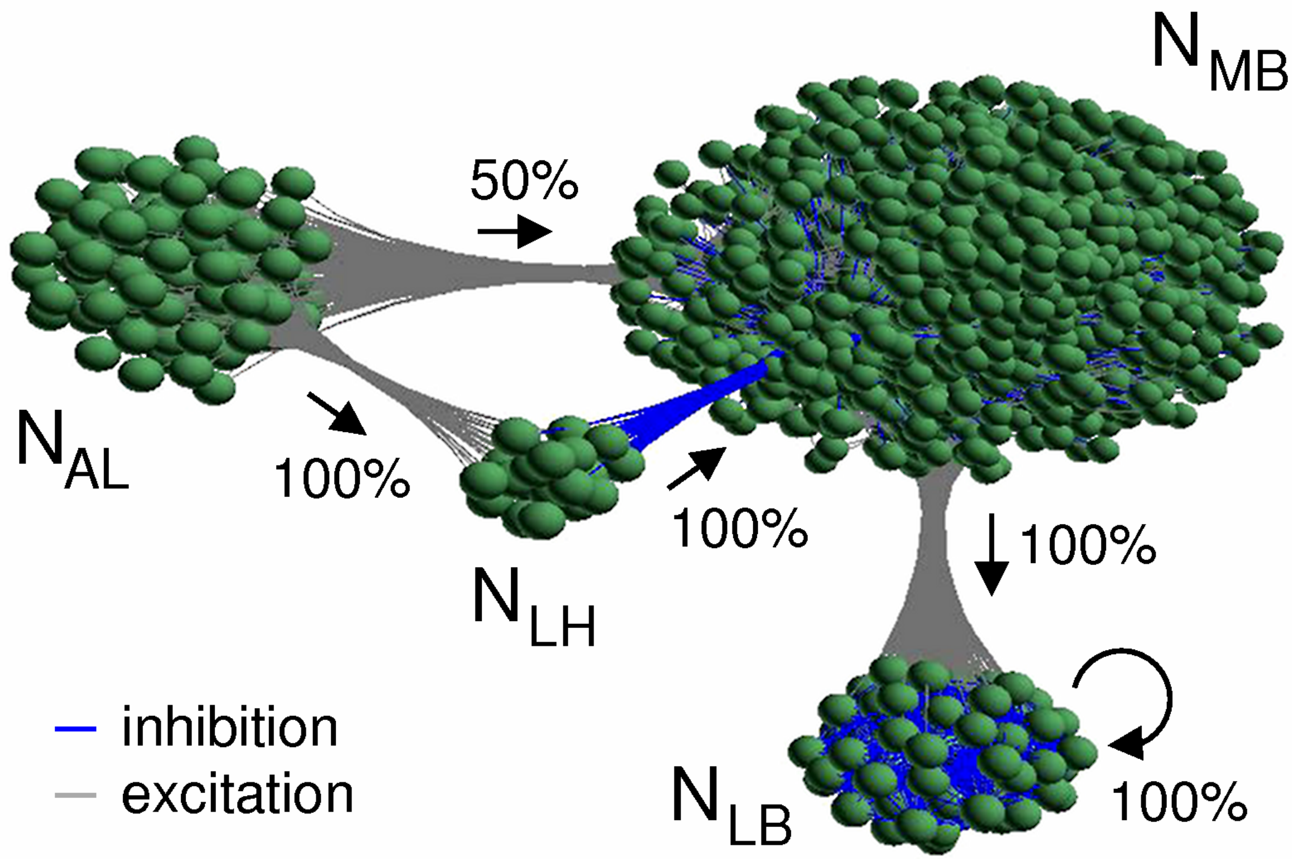
GeNN

GPU enhanced neuronal network simulations

T. Nowotny

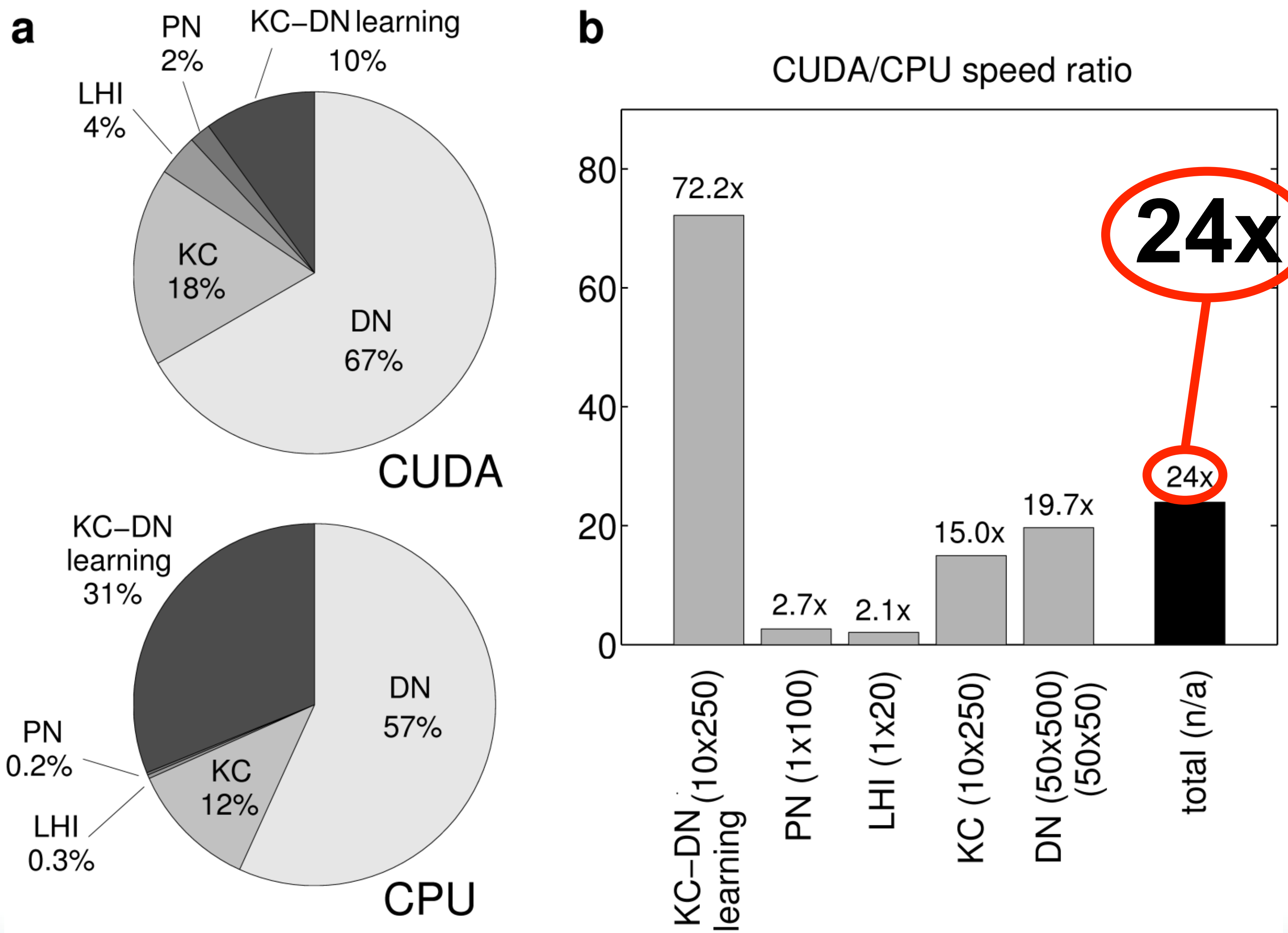
Informatics, CCNR,
University of Sussex

Example: Insect olfaction model



Hand-tuned neuronal network simulation 2009

T. Nowotny, WCCI 2010 Barcelona



However ...

- It took me a month to program a previously developed model (this is after learning how to do CUDA)
- The program was optimised for “my” GPU (a Tesla C870)
- It was optimised for one size of the simulation

But: This code is useless for any other purpose!

Exposure of the challenge: model side

Neural (neuronal) network simulations ...

- Have neurons that could be described as
  - ODEs

$$C\frac{dV}{dt} = \sum_{\mathrm{Na,K,\ldots}} I_{\mathrm{ion}} + \sum_{\mathrm{synapses}} I_{\mathrm{syn}}$$

  - Maps

$$V(t+1) = F(V(t), \{s_i(t)\}, \ldots)$$

  - Stochastic Dynamics

$$p_{\mathrm{fire}}(t) = P(v(t), \{s_i(t)\}, \ldots)$$

Exposure of the challenge: model side

- Neurons interact through synapses that could be
  - Pulse coupling

    $$V_{\text{post}}(t+\Delta t) = V_{\text{post}}(t) + \Delta V$$ if presynaptic spike

  - Instantaneous rise/ exponential decay

    $$\frac{ds}{dt} = -s + \Delta s\, \delta(t - t_{\text{spike}}); \qquad I_{\text{syn}} = gs(V_{\text{rev}} - V_{\text{post}})$$

  - Full-blown ODEs

    $$\dot{r} = \alpha_r F(V_{\text{pre}})(1-r) - \beta_r r$$

    $$\dot{s} = \alpha_s r - \beta_s s$$

    $$I_{\text{syn}} = gs(V_{\text{rev}} - V_{\text{post}})$$

Exposure of the challenge: model side

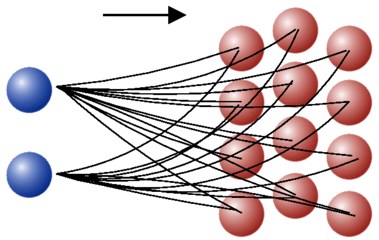
- Synapses can have delays
- Synapses could be plastic

$$\frac{dg}{dt} = F(V_{\mathrm{pre}}, V_{\mathrm{post}}, t)$$

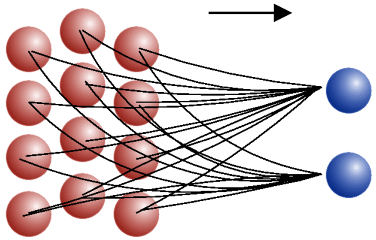
- But note: Spikes are typically rare events: Temporally sparse communication
- The types of neurons, synapses are known at compile time and do not change at runtime

Exposure of the challenge: model side

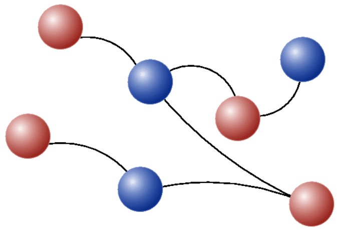
- Synaptic connectivity patterns can be ...
  - Dense pre-synaptically



  - Dense post-synaptically



  - Sparse (pre- or post-synaptically)



  - Local or global, with structure or random

Exposure of the challenge: GPU side

- Need to massively parallelise with minimal communication
  - Could be neuron-based
  - Could be synapses-based
  - ... or both?
  - ... in other reasonable partitions?
- Need to optimise memory access patterns
  - Avoid memory access conflicts
  - Enable coalesced memory access
  - Optimize use of different memory types (register, shared, local, device, textures, …)

Exposure of the challenge: GPU side

- The GPU system used for simulations can
  - Have different numbers of individual GPUs in different configurations (n GPUs on k different boards)
  - have a variety of compute capability levels, in particular different Fixed Point capabilities, atomic operations, etc.
  - have different amounts of memory of each memory type available
  - Have different structures (#multi-processors, #threads/block, #blocks)

Code generation can overcome these problems

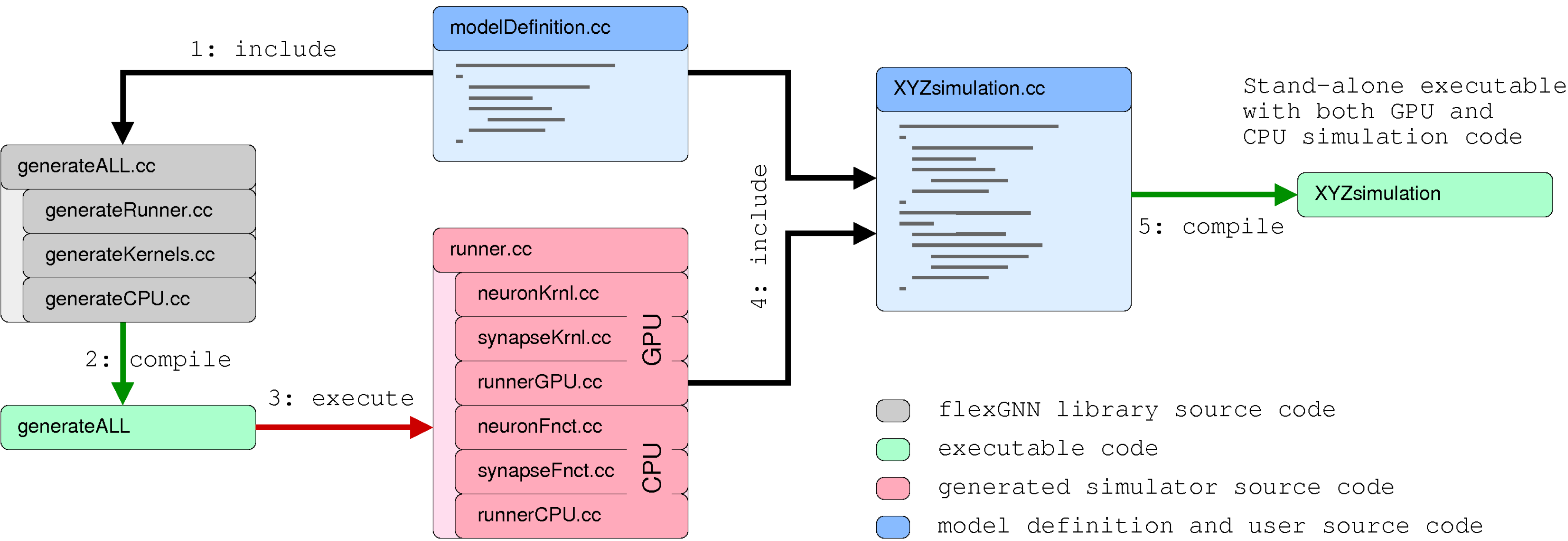
GPU enhanced Neuronal Network simulator provides such a solution:

- Provides a simple C++ API for specifying a neuronal network of interest
- Generates optimised C++ and CUDA code for the model **and for the detected hardware at compile time** (e.g. grid/block organisation, HW capability, model parameters)
- GeNN can offer a large variety of different models – only the ones used in the particular model actually enter the generated code
- The generated code is compiled with the native NVidia compiler (and all its optimisations).

Dr. Thomas Nowotny
CENTRE FOR COMPUTATIONAL NEUROSCIENCE AND ROBOTICS
University of Sussex

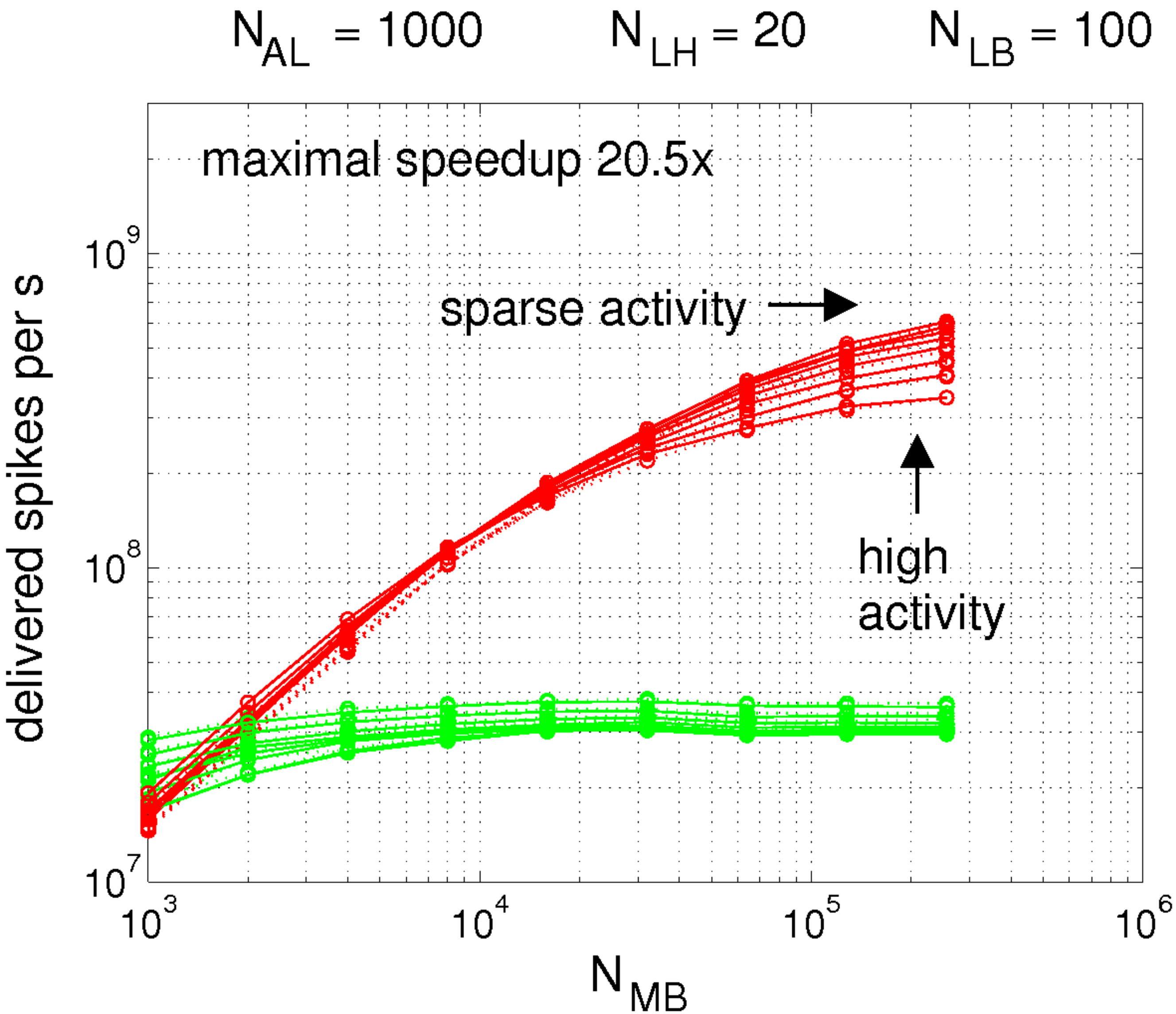
Large & complete

Lean & Mean



Dr. Thomas Nowotny
CENTRE FOR COMPUTATIONAL NEUROSCIENCE AND ROBOTICS
University of Sussex

Performance



AL-MB:
50 % all-to-all

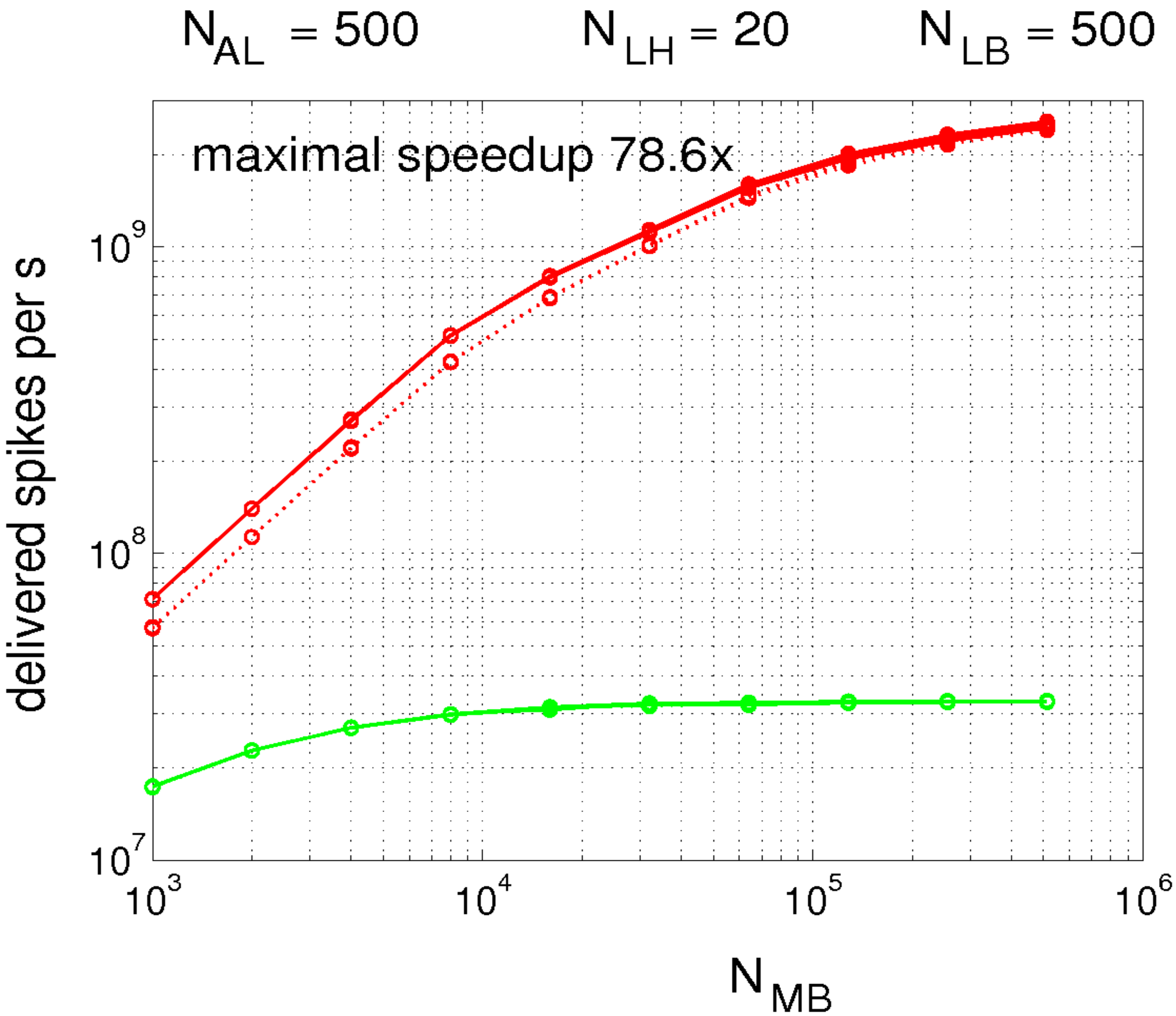
Individual con-
ductances

spikes commu-
nicated to host
(dotted)

spike # commu-
nicated to host
(solid)

Dr. Thomas Nowotny
CENTRE FOR COMPUTATIONAL NEUROSCIENCE AND ROBOTICS
University of Sussex

Performance



AL-MB:
50 % all-to-all

Individual conductances

spikes communicated to host (dotted)

spike # communicated to host (solid)

Conclusions

- Using a C++/CUDA code generation approach has several advantages:
  - Model specific optimisations at compile time
  - Hardware specific optimisations at compile time
  - Can provide unlimited number of different models but actual simulations stay lean and mean
- GeNN is freely extendible with few constraints
- Low level code is accessible if desired/needed
- New hardware capability can be accom-modated

Outlook

- "Mature" the package
  - Extension to CUDA 4, porting to OpenCL
  - More HW-specific optimisations
  - More connectivity pattern optimisations (e.g. Fidjeland "scatter-gather" strategy)
  - Larger library of predefined model elements, delays
- Find a user base & form developer community
- Python API, neuroML API
- Multi chip parallelisation (e.g. many Fermi cards)
- Multi-device parallelisation (p-threads)
- Multi-host parallelisation (mpi) ...

Acknowledgments

- Ramon Huerta
- NVidia professor partnership (2x Quadro FX 5800 cards donated)
- Funders:

BBSRC
bioscience for the future

http://genn.sourceforge.net/