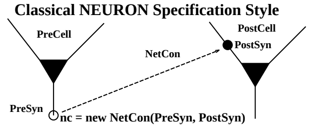
Parallel Network Simulation with NEURON

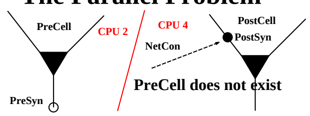
101.4
(PP42)

Michael Hines Yale University, USA NINDS NS11613

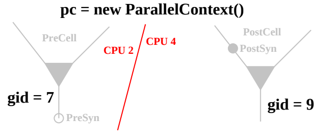
Classical NEURON Specification Style



The Parallel Problem



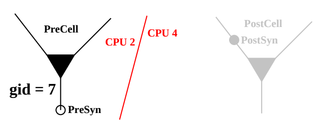
pc = new ParallelContext()



Every spike source (cell) must have a global id number.

```
for (gid = pc.id; gid < ncell; gid += pc.nhost) {
    pc.set_gid2node(gid, pc.id)
}
```

CPU 0		...	CPU 3		CPU 4	
pc.id	0		pc.id	3	pc.id	4
pc.nhost	5		pc.nhost	5	pc.nhost	5
ncell	14		ncell	14	ncell	14
gid			gid		gid	
0			3		4	
5			8		9	
10			13			

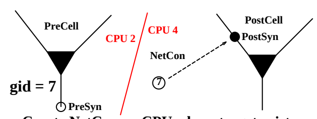


Create cell only where the gid exists.

```
if (pc.gid_exists(7)) {
    PreCell = new Cell()
}
```

Associate gid with spike source.

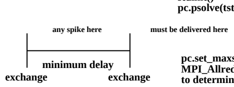
```
nc = new NetCon(PreSyn, nil)
pc.cell(7, nc)
```



Create NetCon on CPU where target exists.

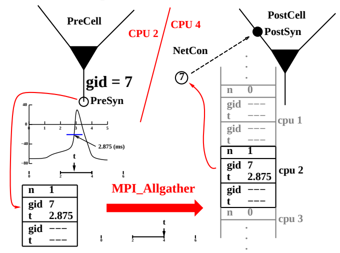
nc = new pc.gid_connect(7, PostSyn)

Run using the idiom pc.set_maxstep(10) stdinit() pc.psolve(tstop)



pc.set_maxstep() uses MPI_Allreduce to determine minimum delay.

Spike Communication



More Efficient Spike Management

Spike exchange buffer compression (Requires fixed step method)

Reduce integration interval to < 256 dt steps, code the double spiketime as a byte.

If there are < 256 cells on each CPU code the int gid as a char local_id.

Select reasonable MPI_Allgather buffer size to send n spikes before requiring an MPI_Allgatherv overflow message.

Bin Queue (Requires fixed step method)

need at least maximum NetCon delay / dt bins

ARTIFICIAL_CELL SelfEvents bypass queue

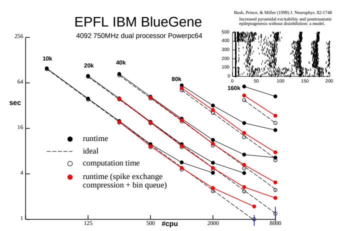
(Requires the integration interval be <= the positive global minimum NetCon delay)

On every incoming NetCon event check to see if SelfEvent < t

After each integration interval iterate over outstanding SelfEvents to deliver all that are < t.

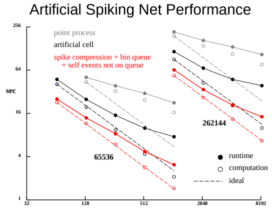
EPFL IBM BlueGene

4092 750MHz dual processor Powerpc64



#cells	#states	#netcons	#outputSpikes	#spikedeliver
10k	444,664	17,167,785	31,118	52,040,794
20k	888,664	68,655,566	33,960	110,634,002
40k	1,777,664	274,591,128	69,529	452,987,907
80k	3,553,664	1,098,302,267	294974	2,147,483,647
160k	7,107,664	4,393,084,577	844,175	22,847,784,937

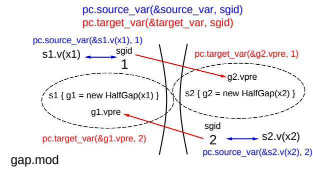
Artificial Spiking Net Performance



Each cell fires randomly every 10 to 20 ms.
65K cells, 1000 random connections per cell
256K cells, 10,000 random connections per cell

tstop = 200(ms)
delay = 1(ms)
weight = 0

Continuous Voltage Exchange

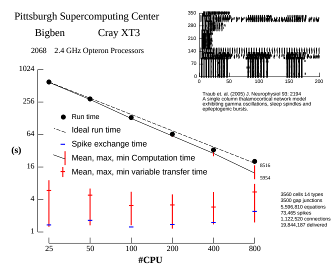


gap.mod

```
NEURON {
  POINT_PROCESS HalfGap
  ELECTRODE_CURRENT i
  RANGE r, i, vgap
}
PARAMETER { r = 1e9 (megohm) }

ASSIGNED {
  v (millivolt)
  vgap (millivolt)
  i (nanoamp)
}
BREAKPOINT { i = (vgap - v)/r }
```

Pittsburgh Supercomputing Center
Bigben Cray XT3
2068 2.4 GHz Opteron Processors



Acknowledgements

Supported by NINDS NS11613

Henry Markram for the use of the EPFL IBM BlueGene and development support under the auspices of the Blue Brain project.

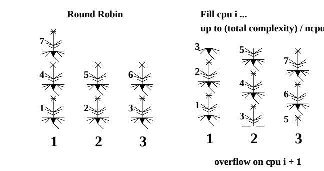
Roger Traub for making his model available in ModelDB.

Terry Sejnowski for the use of the SDSC DataStar.

Pittsburgh Supercomputer Center for the use of the Cray XT3.

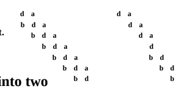
Load Balance

7 cells 3 cpus (or heterogeneous cells)

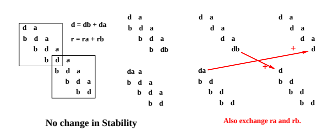


but... what is the overhead of splitting a cell?

Optimal Gaussian elimination:
Triangularize from leaves to root.

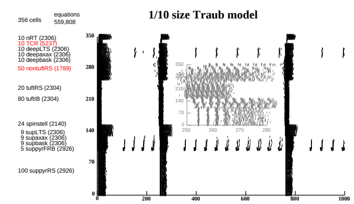


Any tree can be split into two subtrees with a shared root node.



No change in Stability
Accuracy
Complexity

1/10 size Traub model



ncpu	avgload	unbalanced maxload		balanced maxload	
32	40584	42870	6%	40963	0%
64	20292	23310	15%	20496	1%
128	10146	13350	32%	10260	1%
256	5073	9461	86%	5237	3%
512	2536	5475	116%	2839	11%