Parallel Computation

"Faster" is the only reason

- But...
  - greater programming complexity
  - new kinds of bugs
    - ...and not much help for fixing them.

Parallel Computation

"Faster" is the only reason

But...

greater programming complexity

new kinds of bugs

...and not much help for fixing them.

Can the day or week of user effort be recovered?

Parallel Computation

"Faster" is the only reason

But...

greater programming complexity

new kinds of bugs

...and not much help for fixing them.

Can the day or week of user effort be recovered?

8192 processor EPFL IBM BlueGene

1 hour at 700MHz

3 months at 3GHz

Parallel Computation

A simulation run takes about a second

want to do 1000's of them,

varying a dozen or so parameters.

A simulation run takes hours.

want to spread the problem over several machines.

Parallel Computation

A simulation run takes about a second

want to do 1000's of them,

varying a dozen or so parameters.

- Screensaver Calin−Jageman and Katz, 2006
- Bulletin−board (Linda)

A simulation run takes hours.

want to spread the problem over several machines.

Parallel Computation

A simulation run takes about a second

want to do 1000’s of them,

varying a dozen or so parameters.

A simulation run takes hours.

want to spread the problem over several machines.

Parallel Computation

A simulation run takes hours.

want to spread the problem over several machines.

Network

Subnets on different machines

Cells communicate by:

logical spike events with significant axonal, synaptic delay.

postsynaptic conductance depends continuously on presynaptic voltage.

gap junctions

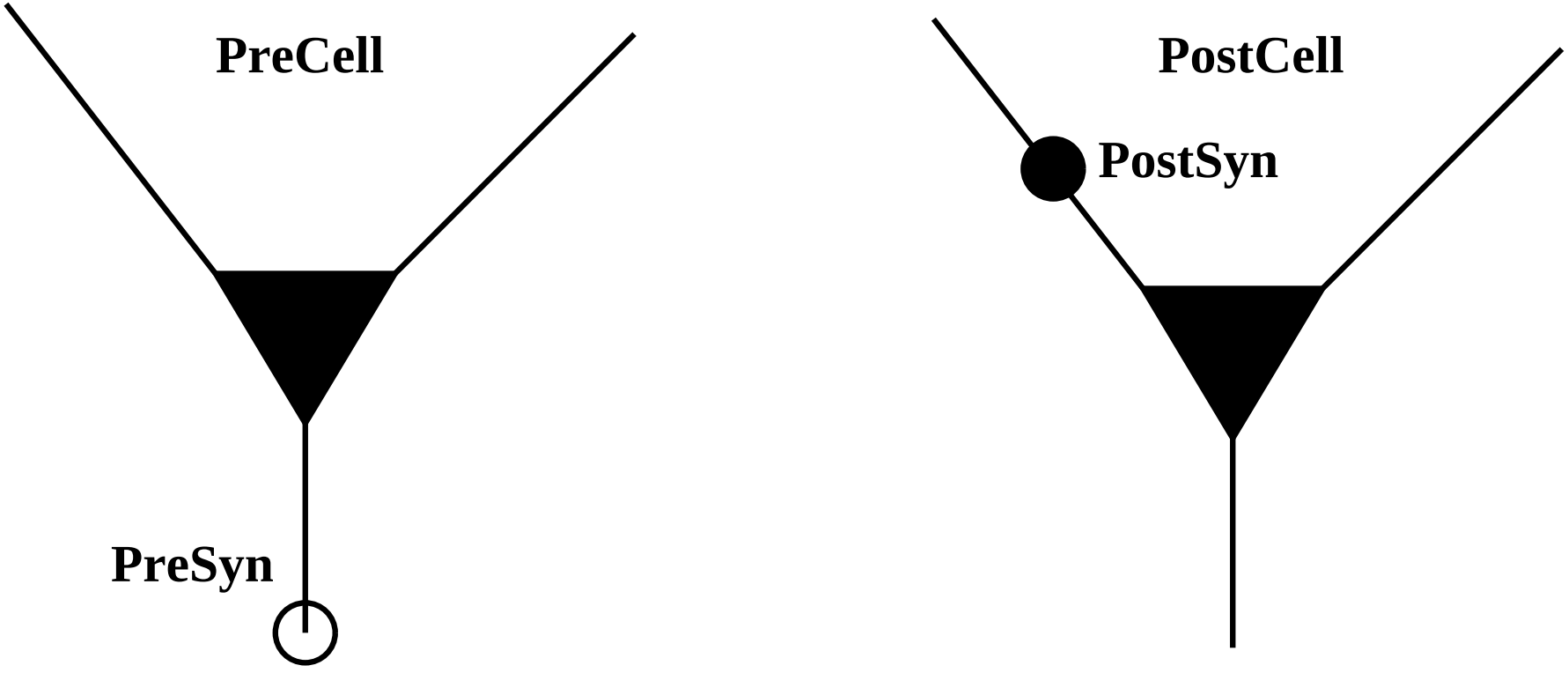
Parallel Computation

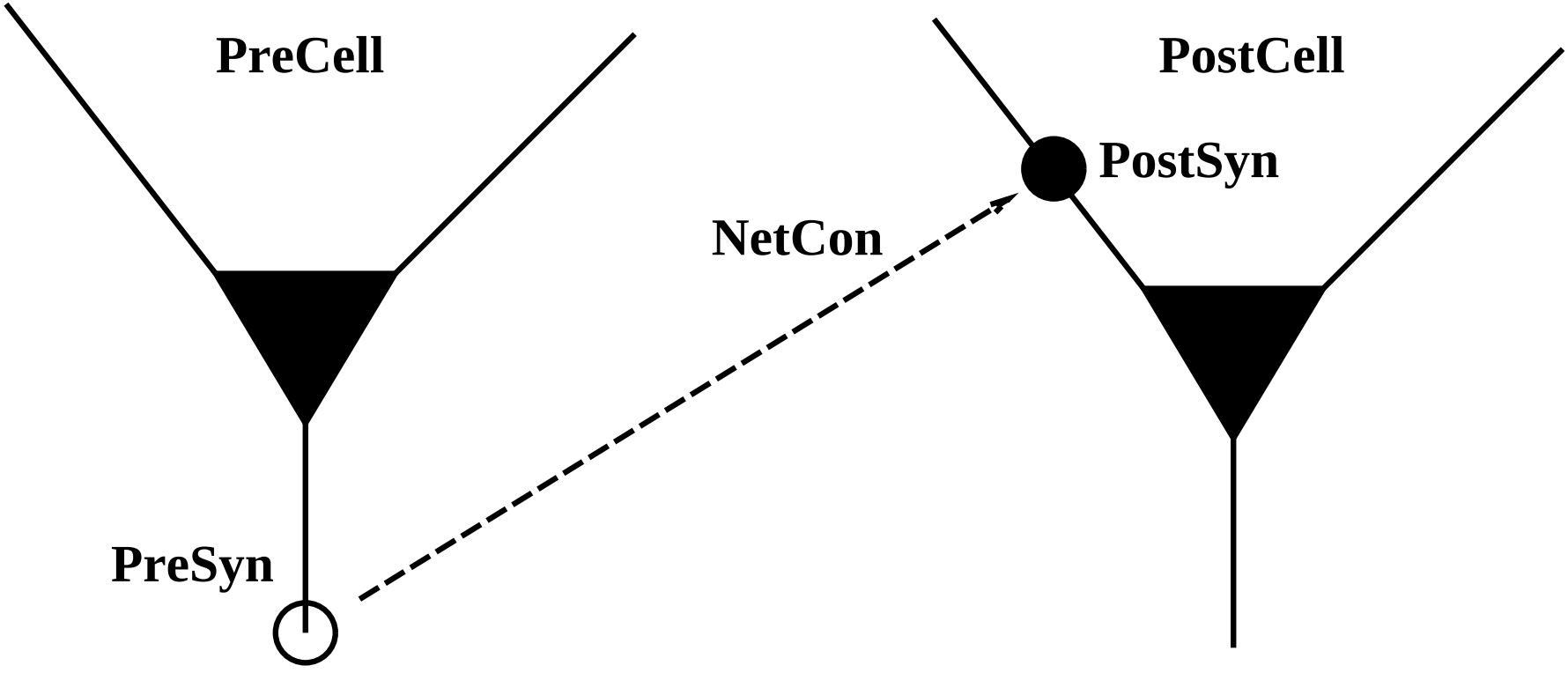
A simulation run takes hours.

want to spread the problem over several machines.

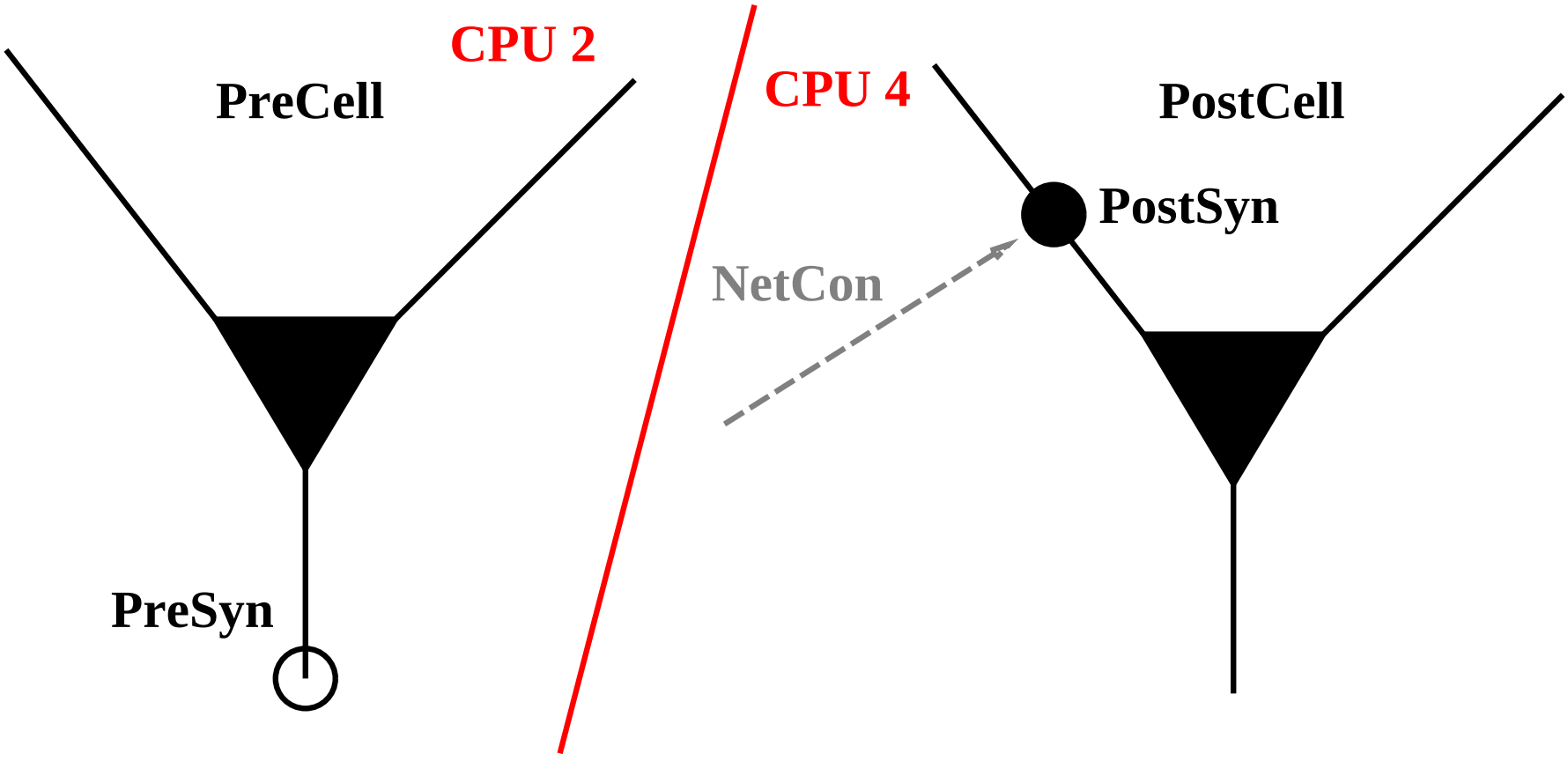
Single cells

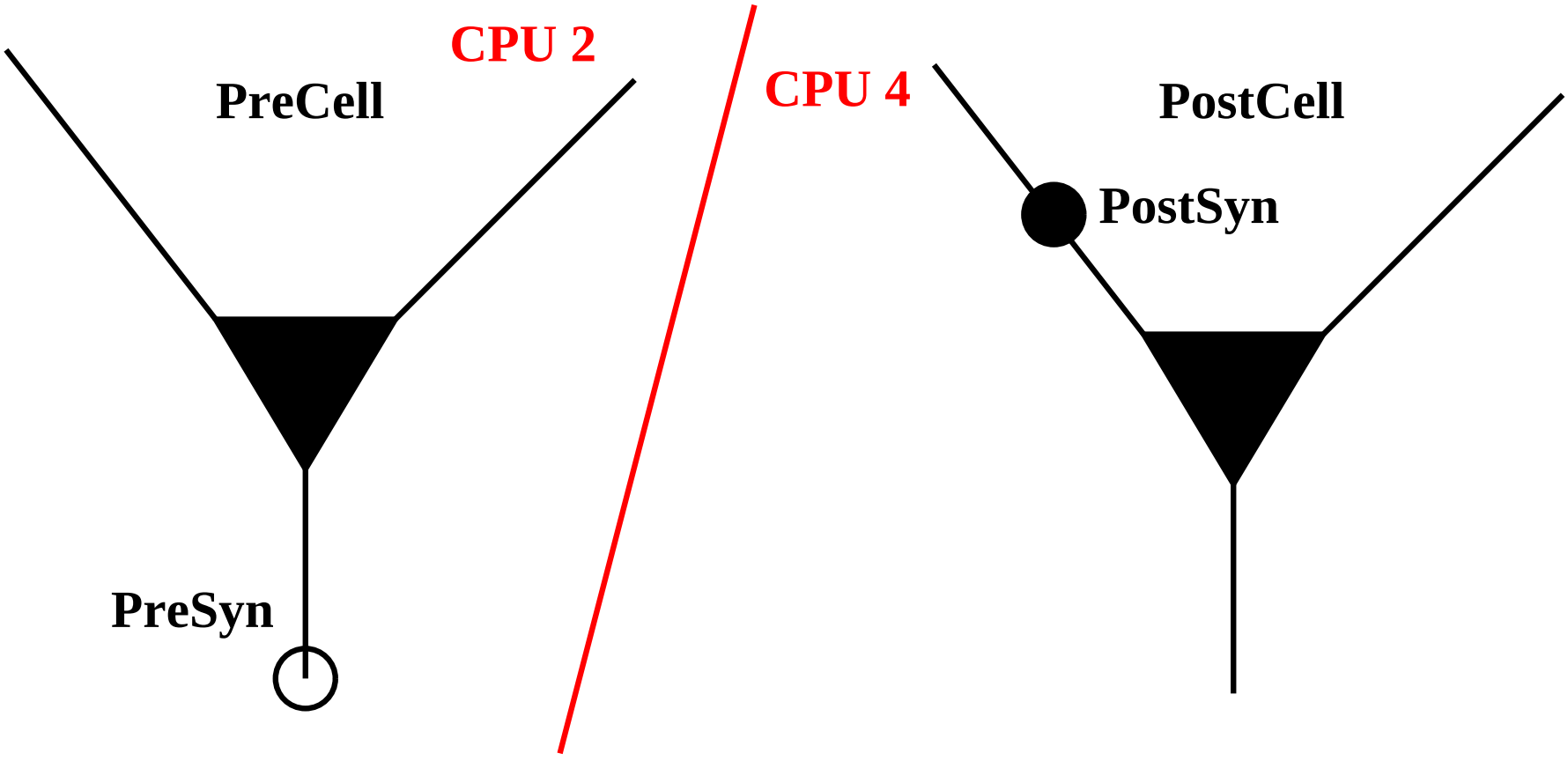
portions of the tree cable equation on different machines.


PreCell
PreSyn
PostCell
PostSyn

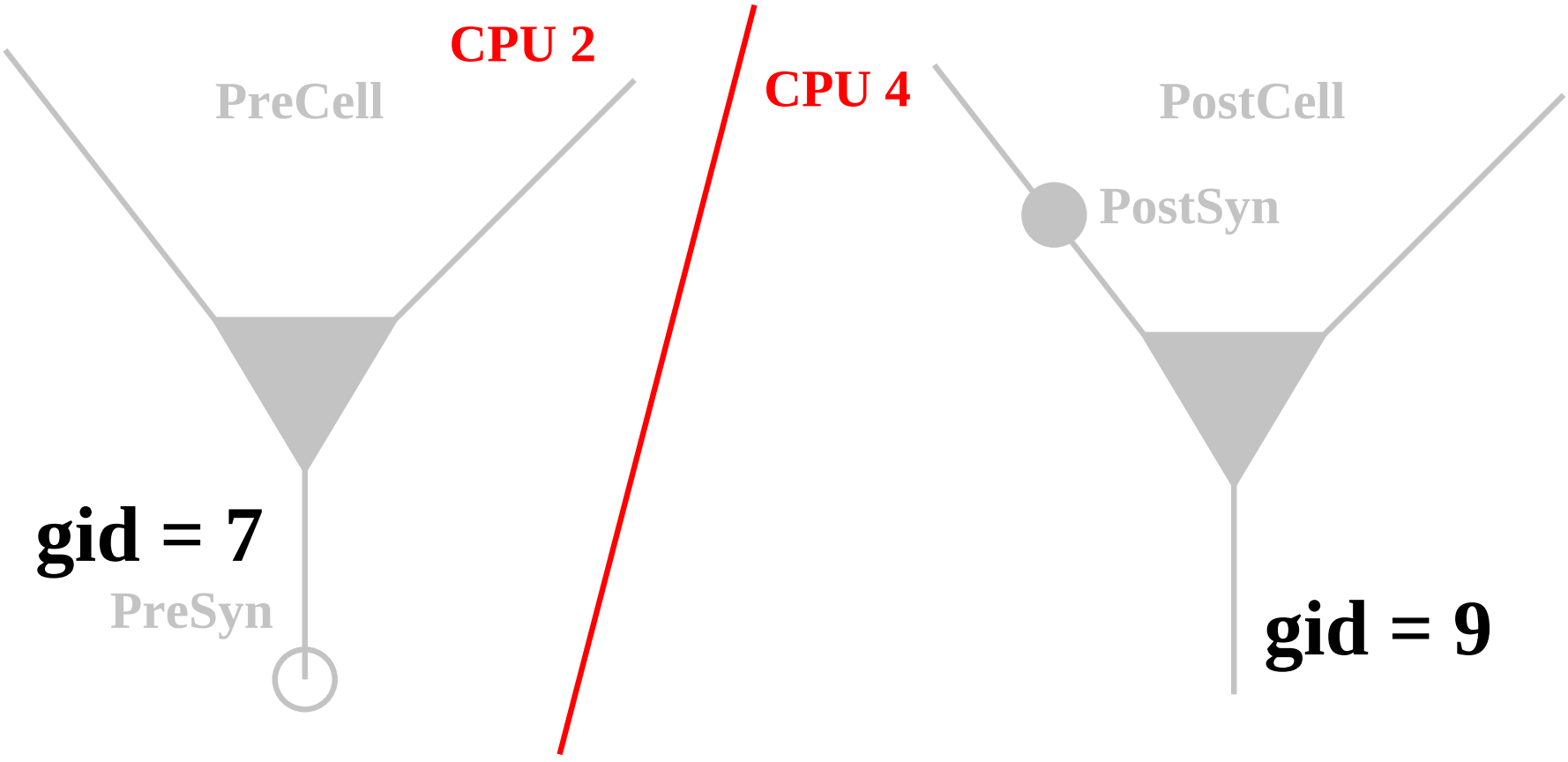



```
nc = new NetCon(PreSyn, PostSyn)
```


CPU 2
PreCell
CPU 4
PostCell
PostSyn
NetCon
PreSyn




```
pc = new ParallelContext()
```



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

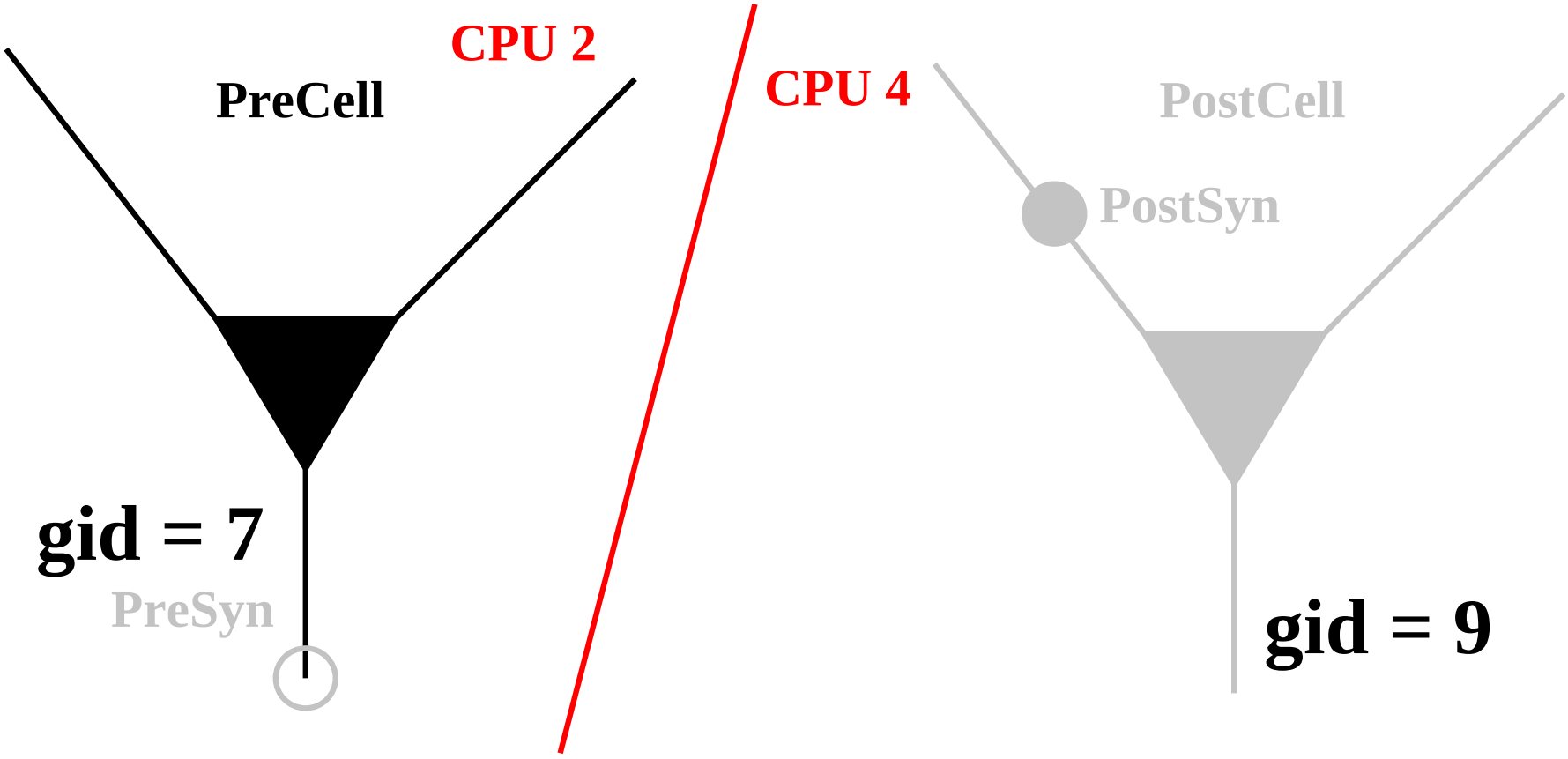
CPU 0		CPU 3		CPU 4	
pc.id	0	pc.id	3	pc.id	4
pc.nhost	5	pc.nhost	5	pc.nhost	5
ncell	14	ncell	14	ncell	14
gid		gid		gid	
0		3		4	
5		8		9	
10		13			

...

An efficient way to distribute:

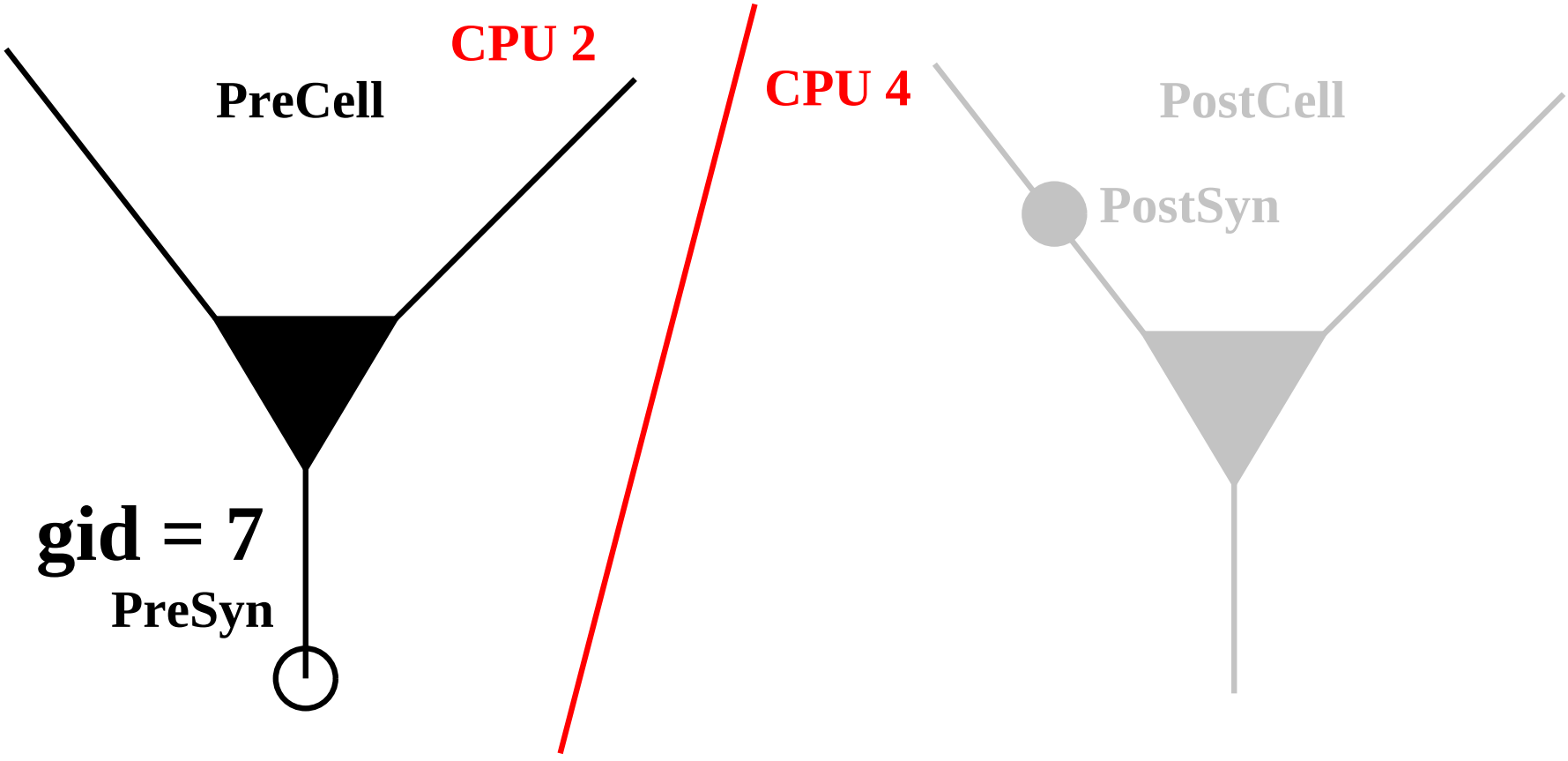
```
for (gid = pc.id; gid < ncell; gid += pc.nhost) {
    pc.set_gid2node(gid, pc.id)
    ...
}
```

body executed only ncell/nhost times, not ncell.



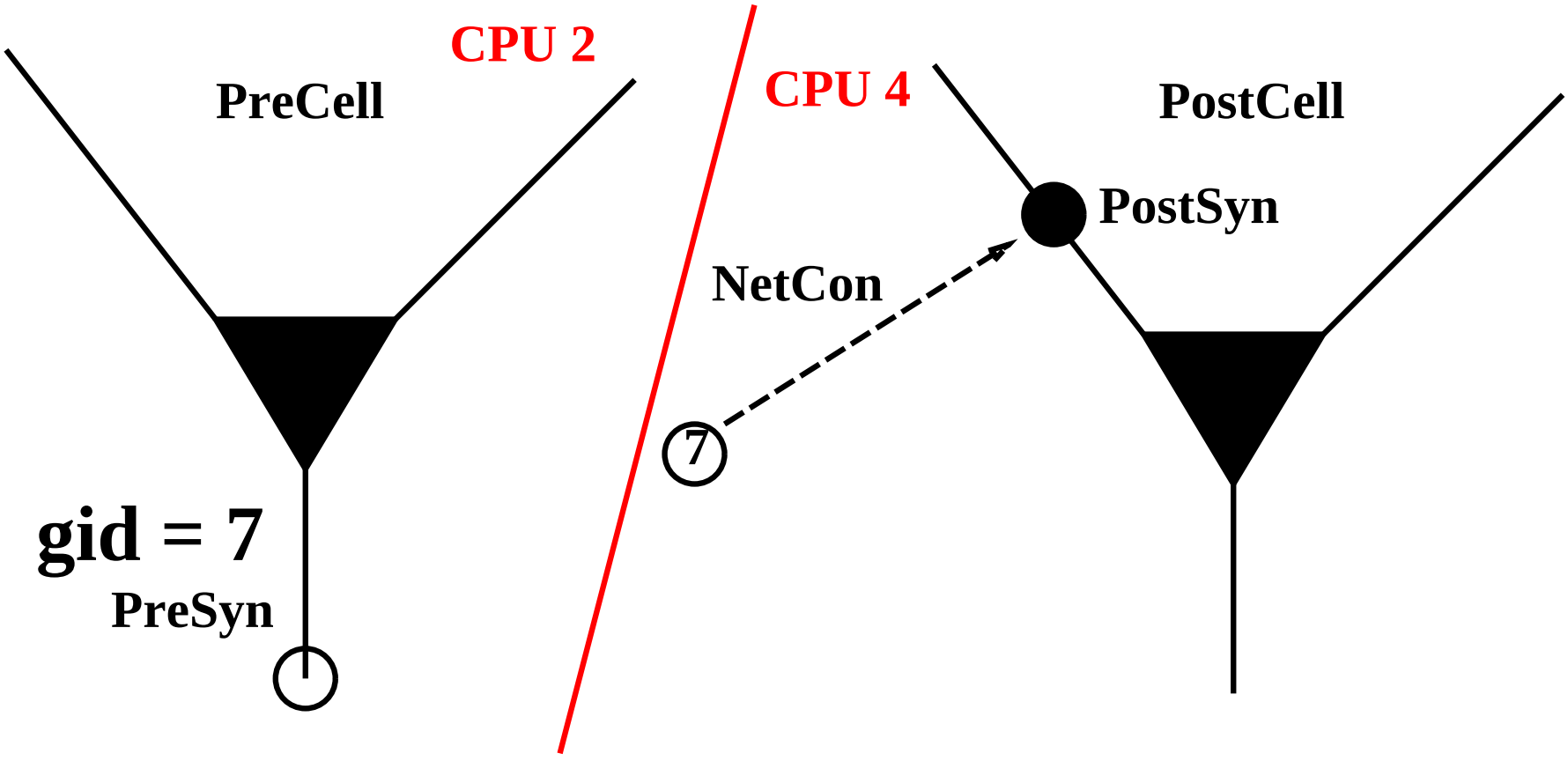
Create cell only where the gid exists.

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



Associate gid with spike source.

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



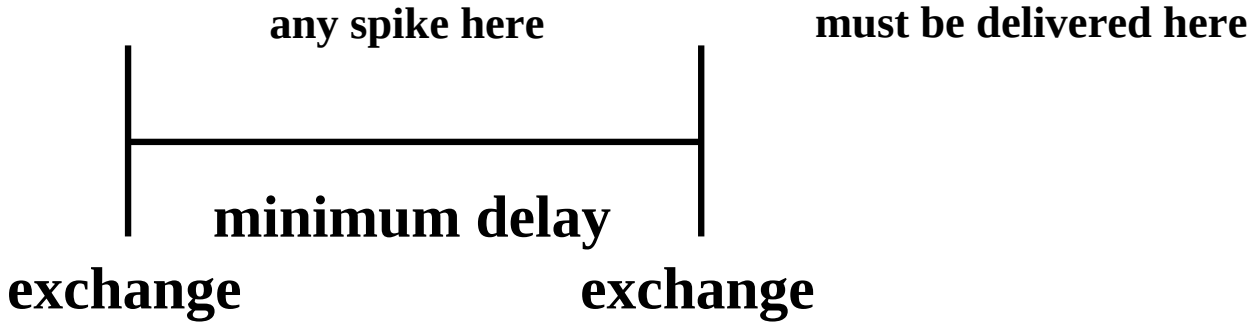
Create NetCon on CPU where target exists.

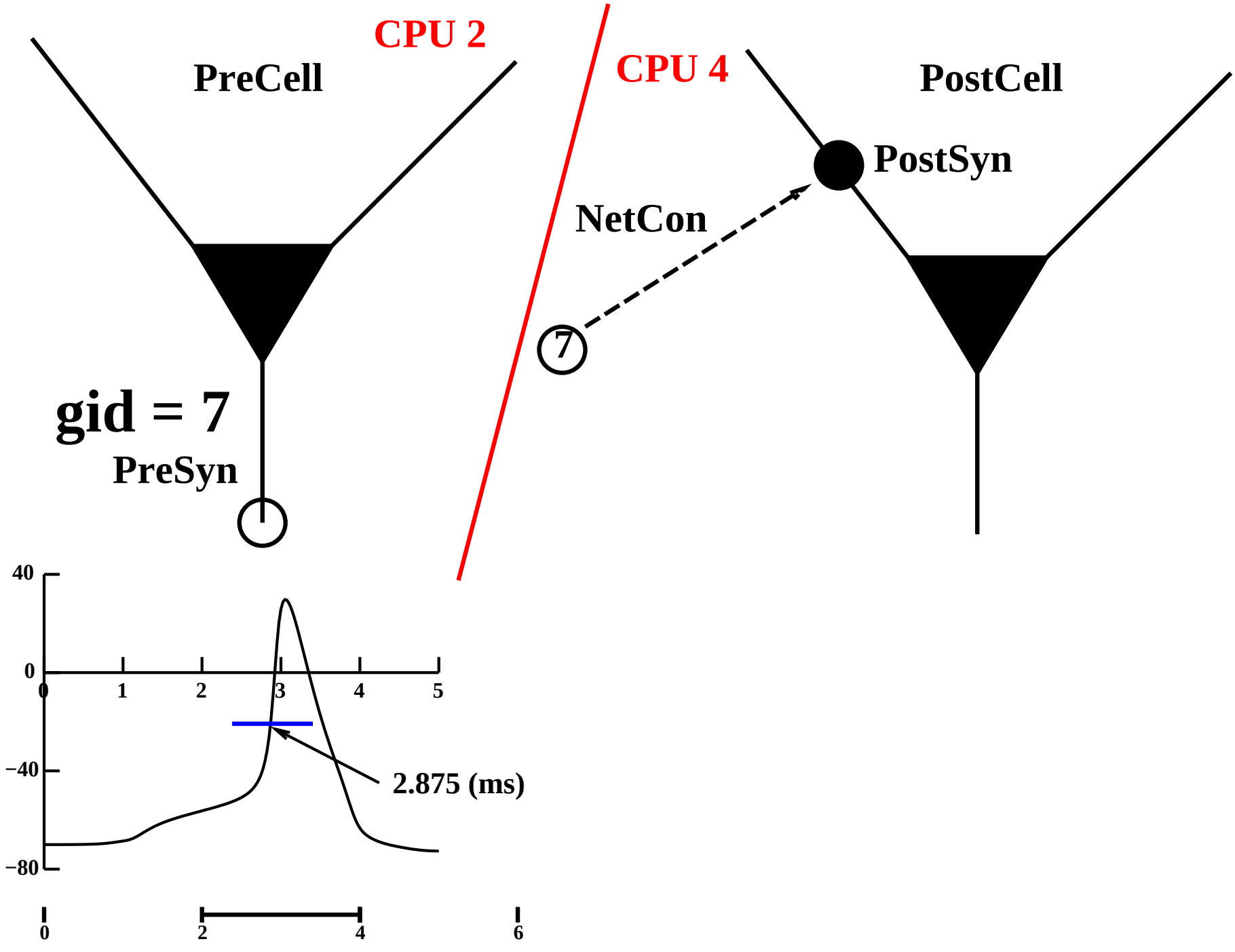
```
nc = pc.gid_connect(7, PostSyn)
```

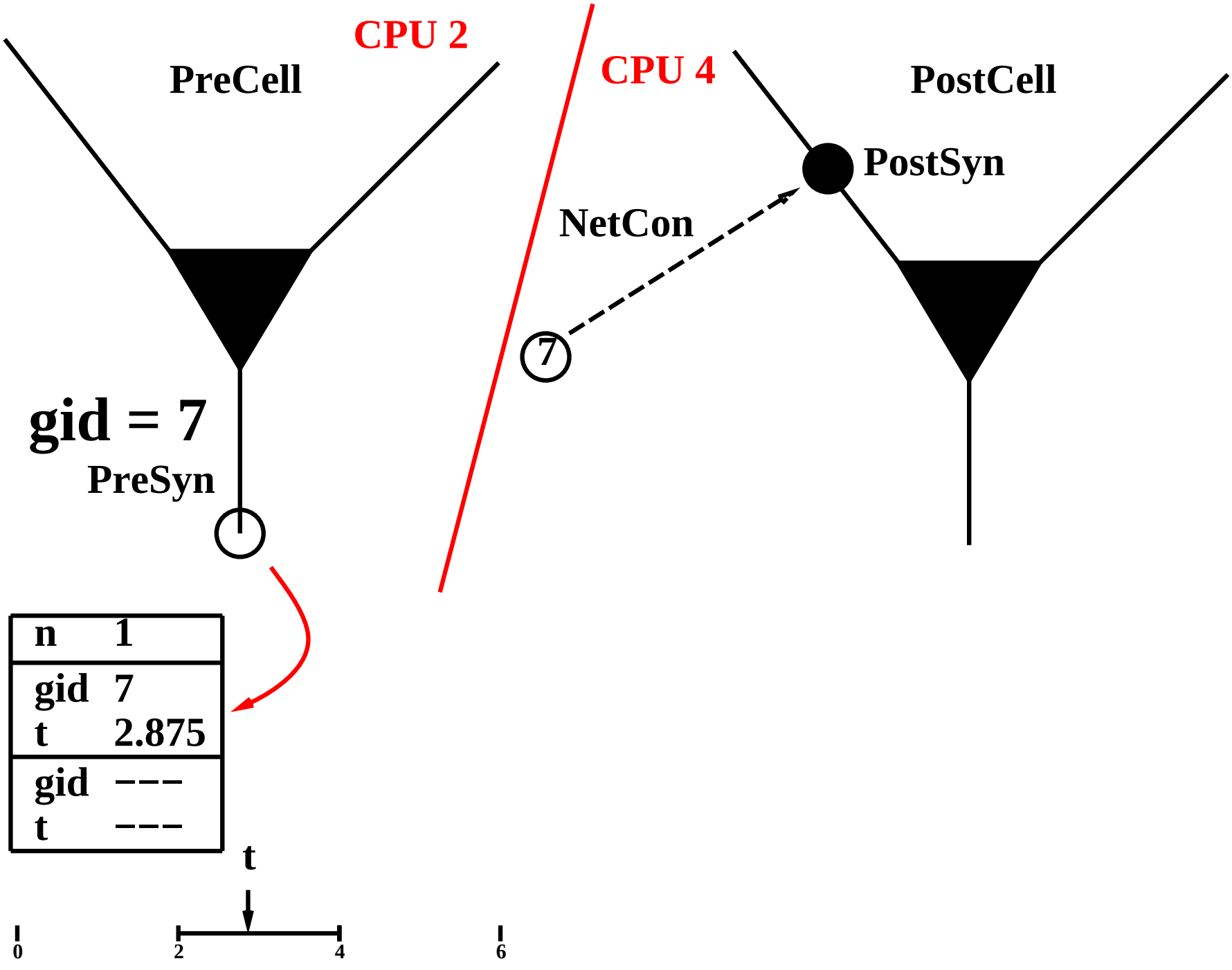
Run using the idiom

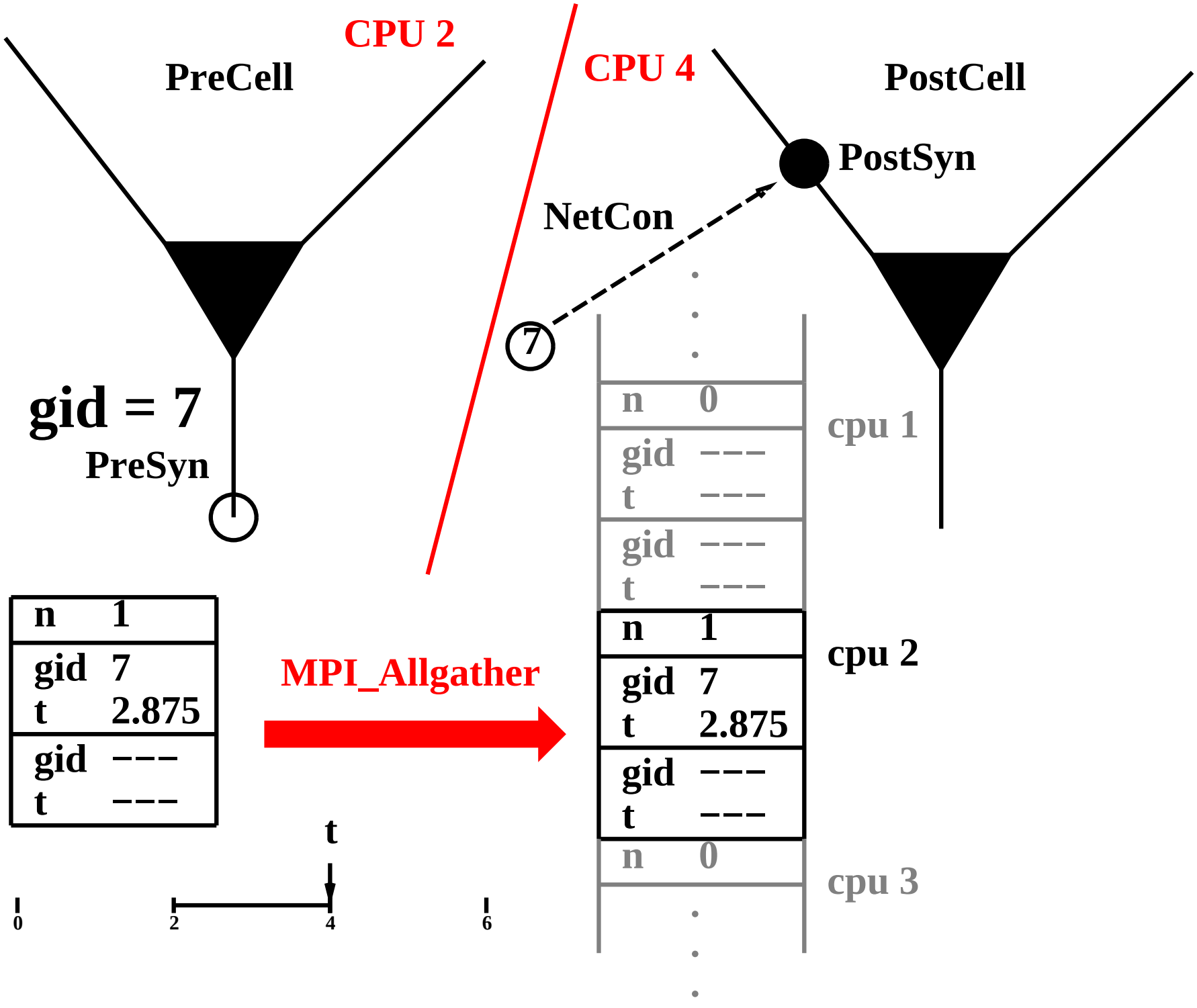
```
pc.set_maxstep(10)
stdinit()
pc.psolve(tstop)
```

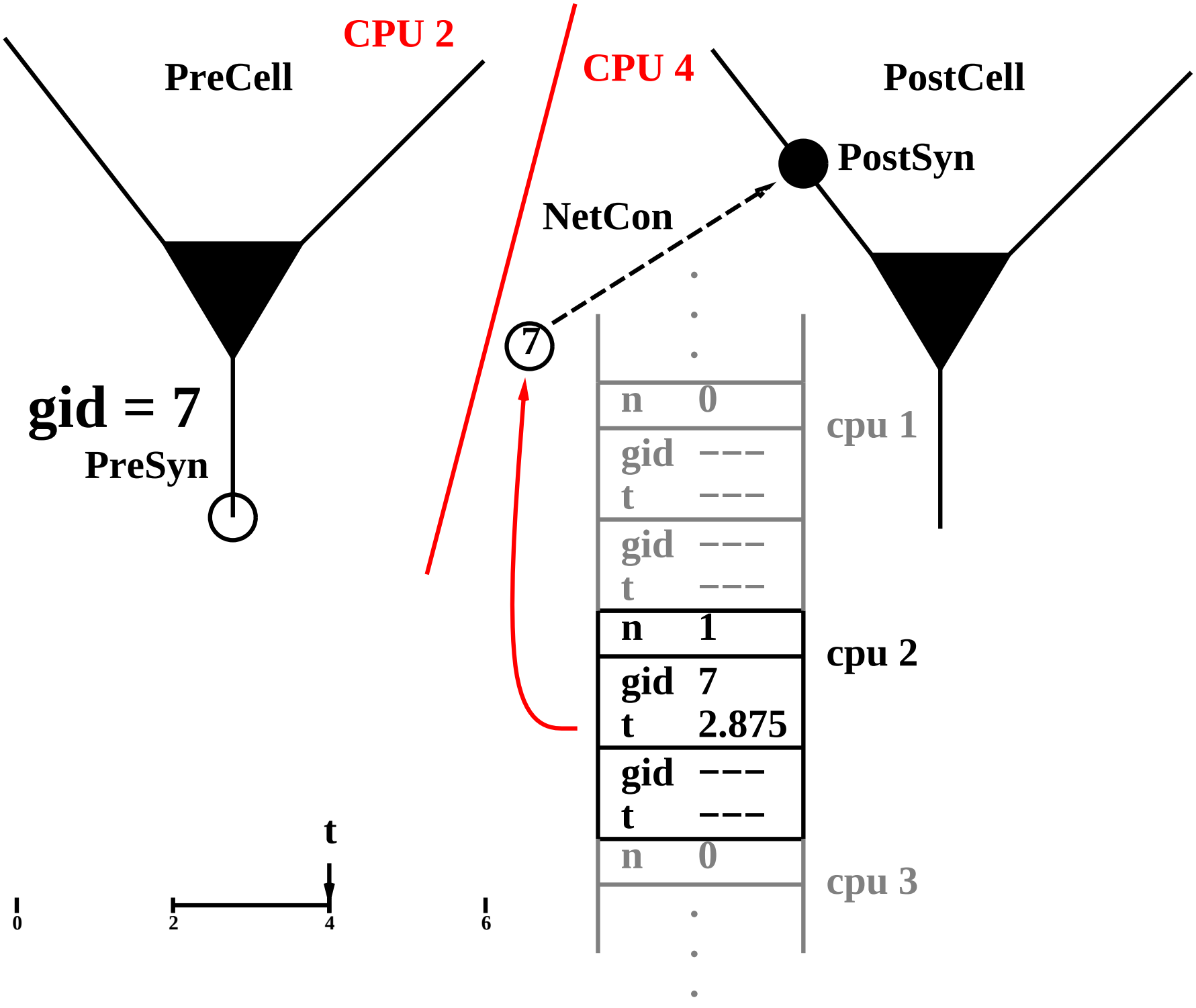
pc.set_maxstep() uses MPI_Allreduce to determine minimum delay.



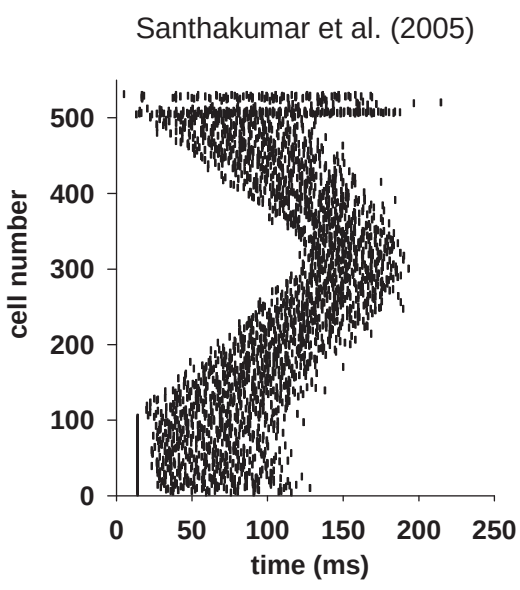

CPU 2
PreCell
CPU 4
PostCell
PostSyn
NetCon
7
gid = 7
PreSyn
40
0
−40
−80
0
1
2
3
4
5
2.875 (ms)
0
2
4
6

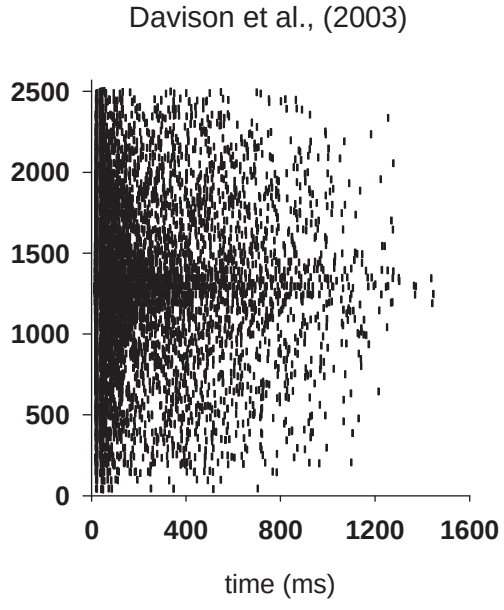

CPU 2
CPU 4
PreCell
PostCell
PostSyn
NetCon
7
gid = 7
PreSyn
n 1
gid 7
t 2.875
gid ---
t ---
t
0
2
4
6

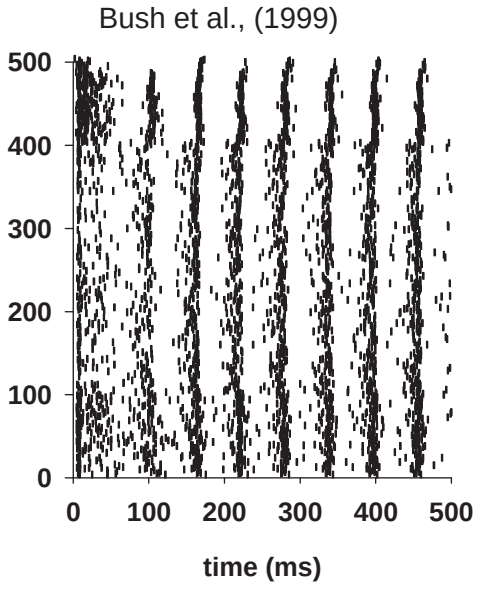

CPU 2
CPU 4
PreCell
PostCell
PostSyn
NetCon
7
gid = 7
PreSyn
n 1
gid 7
t 2.875
gid ---
t ---
MPI_Allgather
t
0
2
4
6
n 0
gid ---
t ---
gid ---
t ---
n 1
gid 7
t 2.875
gid ---
t ---
n 0
cpu 1
cpu 2
cpu 3

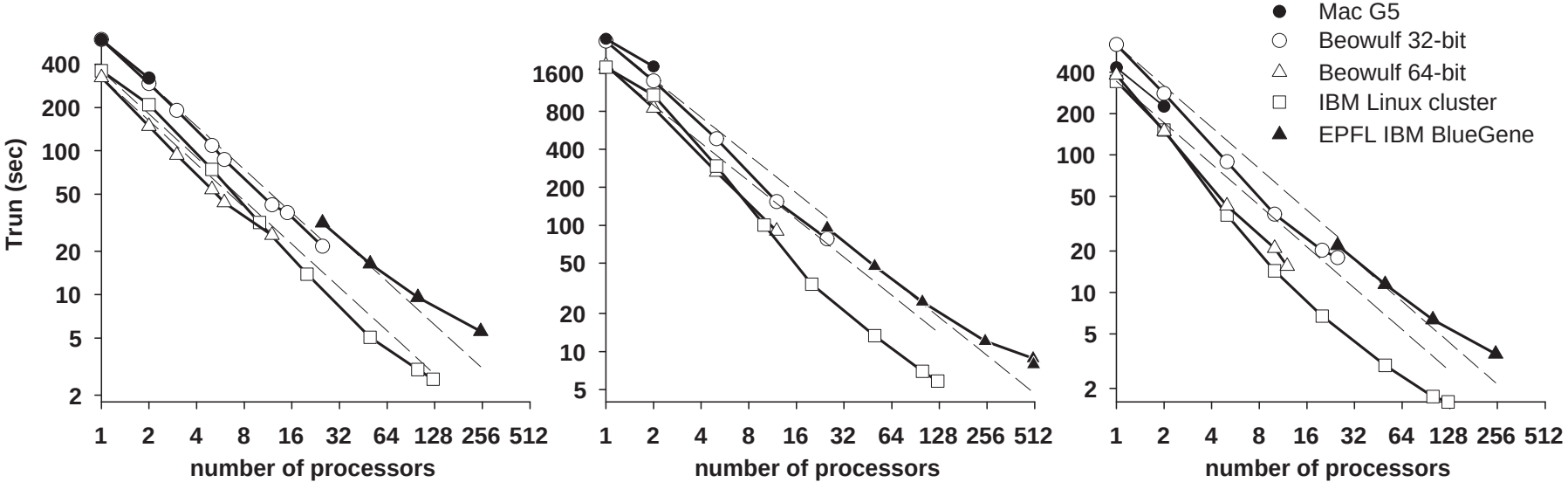

CPU 2
PreCell
CPU 4
PostCell
PostSyn
NetCon
7
gid = 7
PreSyn
n 0
gid ---
t ---
gid ---
t ---
n 1
gid 7
t 2.875
gid ---
t ---
n 0
cpu 1
cpu 2
cpu 3
t
0
2
4
6

Migliore et al (2006) J. Comput. Neurosci. 21(2):119



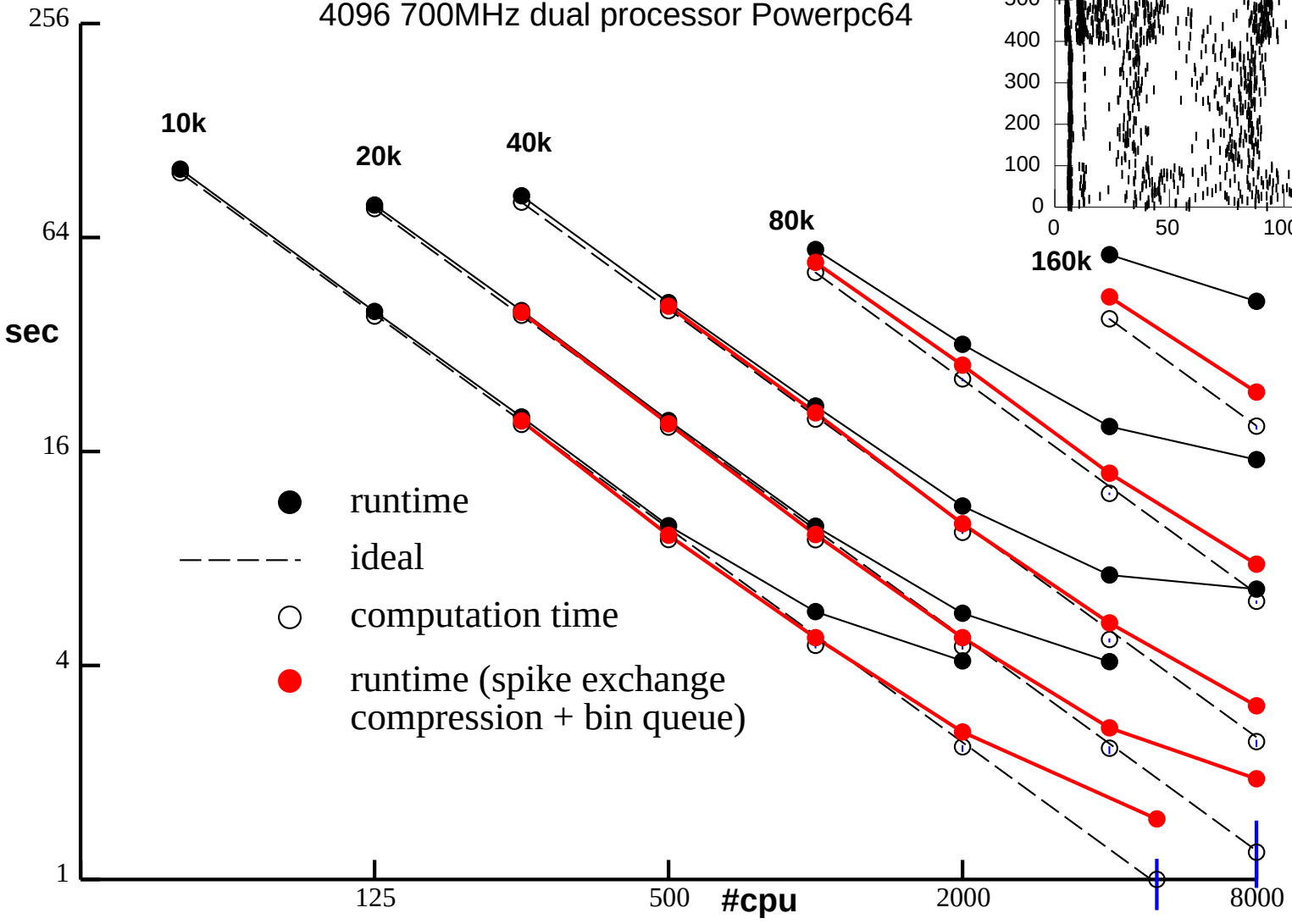






EPFL IBM BlueGene

4096 700MHz dual processor Powerpc64

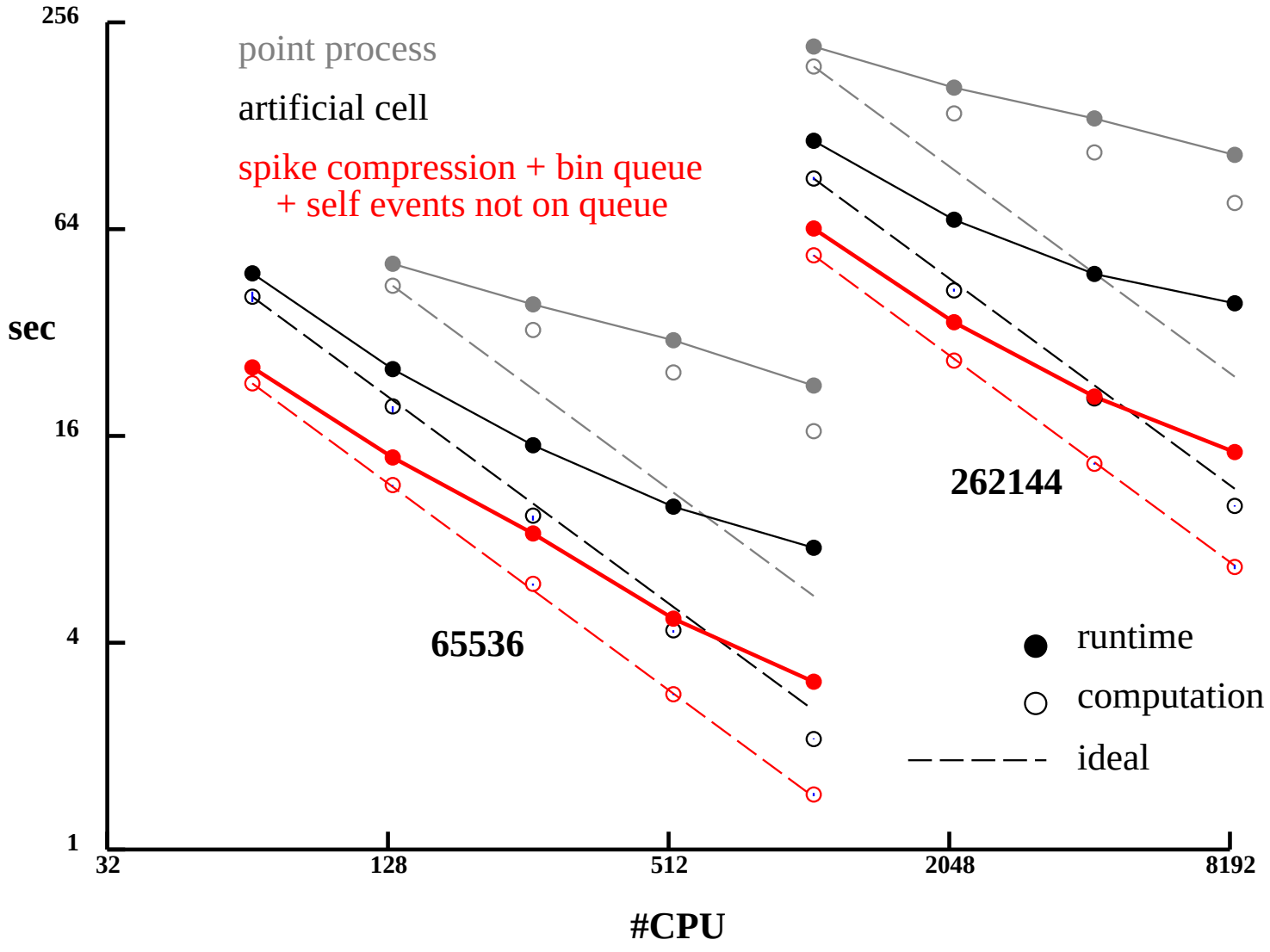


Bush, Prince, & Miller (1999) J. Neurophys. 82:1748

Increased pyramidal excitability and posttraumatic epileptogenesis without disinhibition: a model.

#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

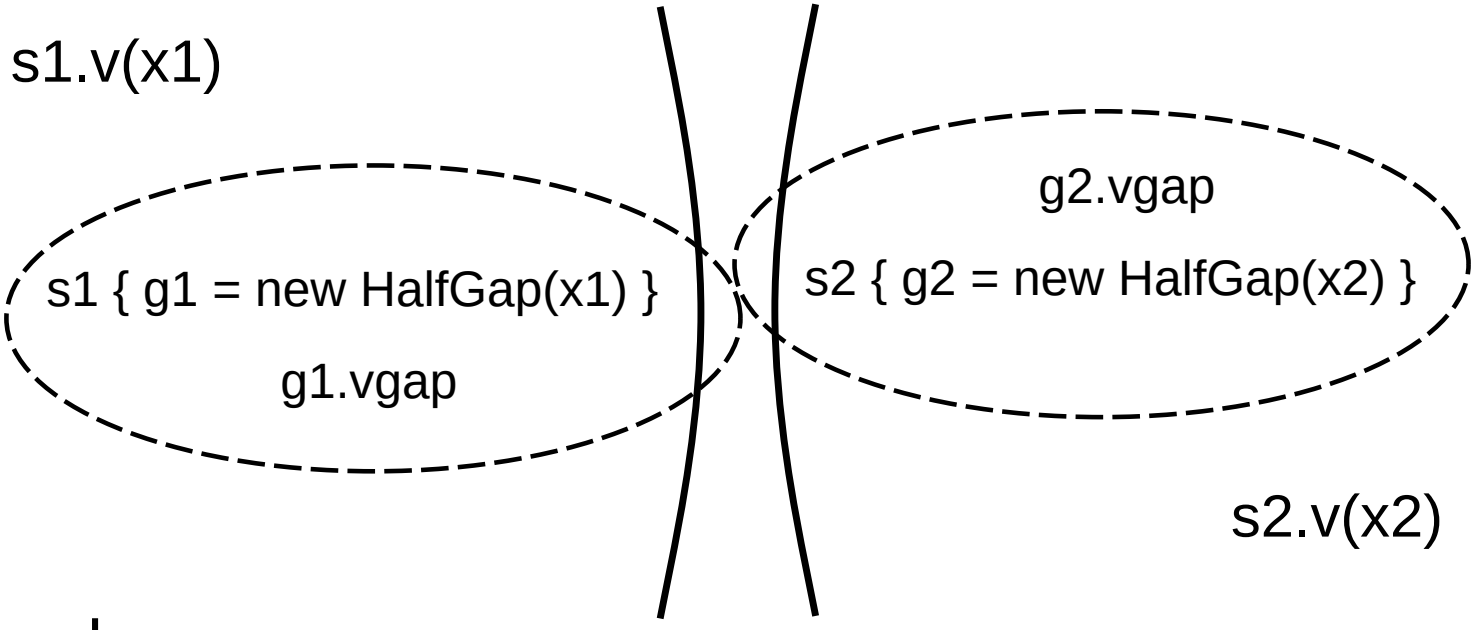
Artificial Spiking Net Performance



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



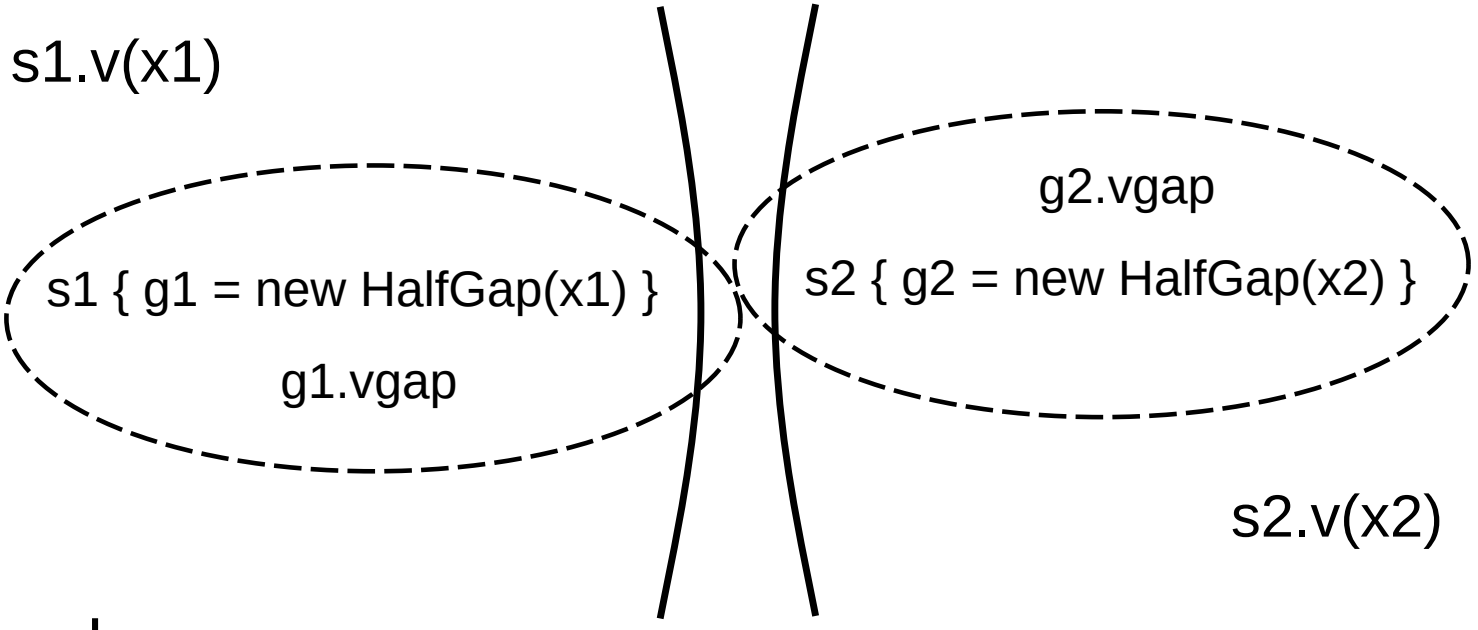
gap.mod

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

```
ASSIGNED {
  v (millivolt)
  vgap (millivolt)
  i (nanoamp)
}
CURRENT { i = (vgap − v)/r }
```

Continuous Voltage Exchange

pc.source_var(&source_var, sgid)



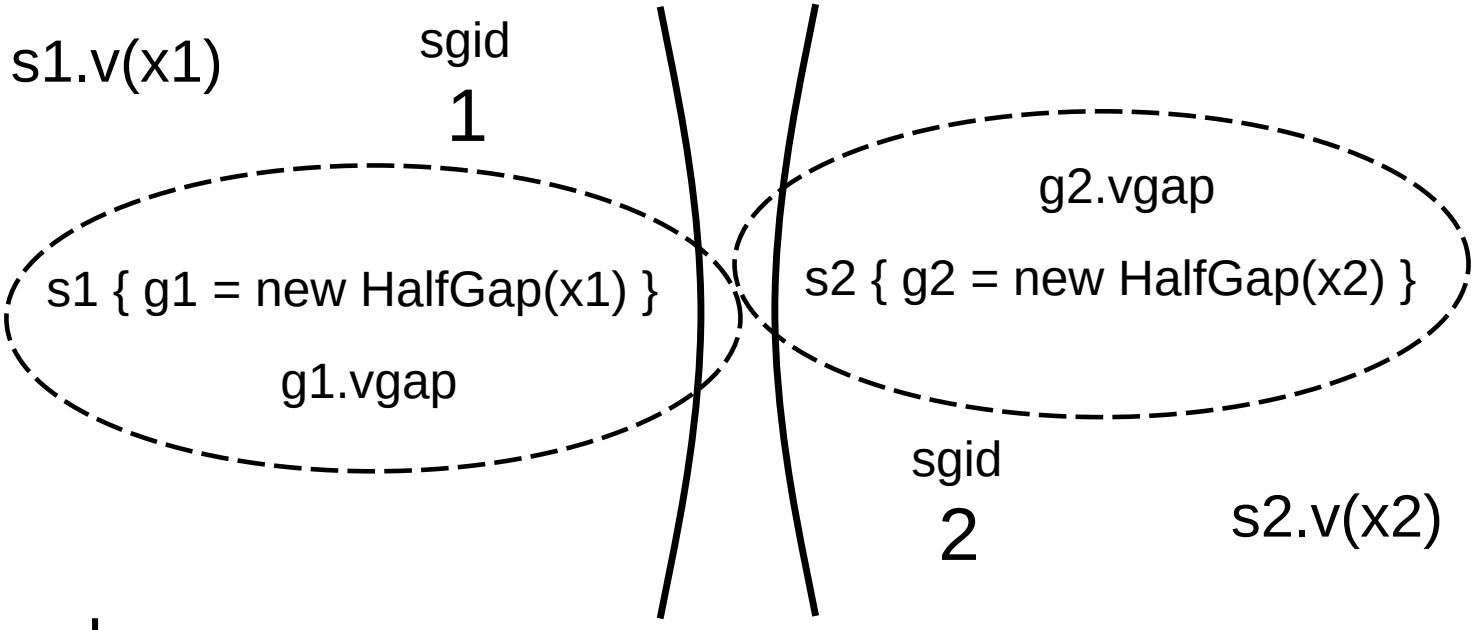
gap.mod

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

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

Continuous Voltage Exchange

pc.source_var(&source_var, sgid)



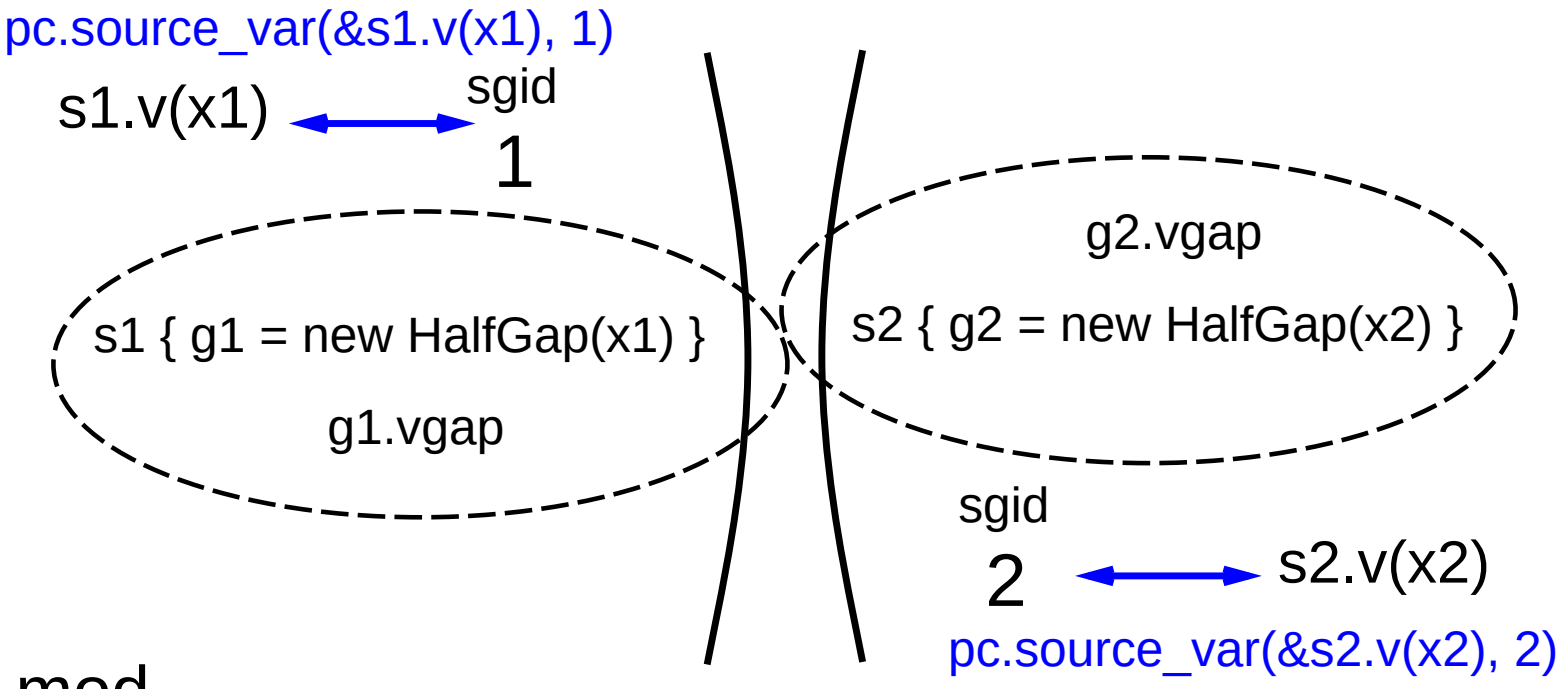
gap.mod

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

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

Continuous Voltage Exchange

pc.source_var(&source_var, sgid)



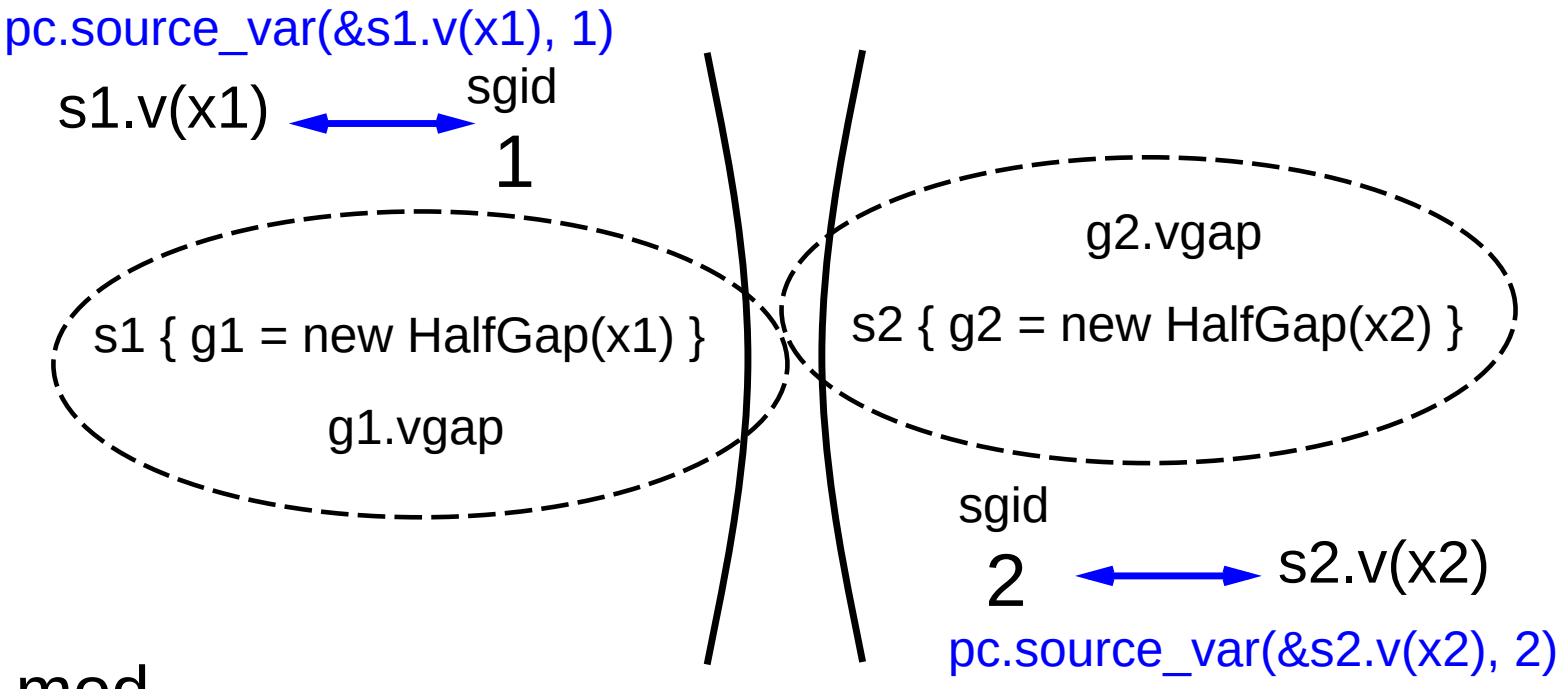
gap.mod

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

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

Continuous Voltage Exchange

pc.source_var(&source_var, sgid)
pc.target_var(&target_var, sgid)



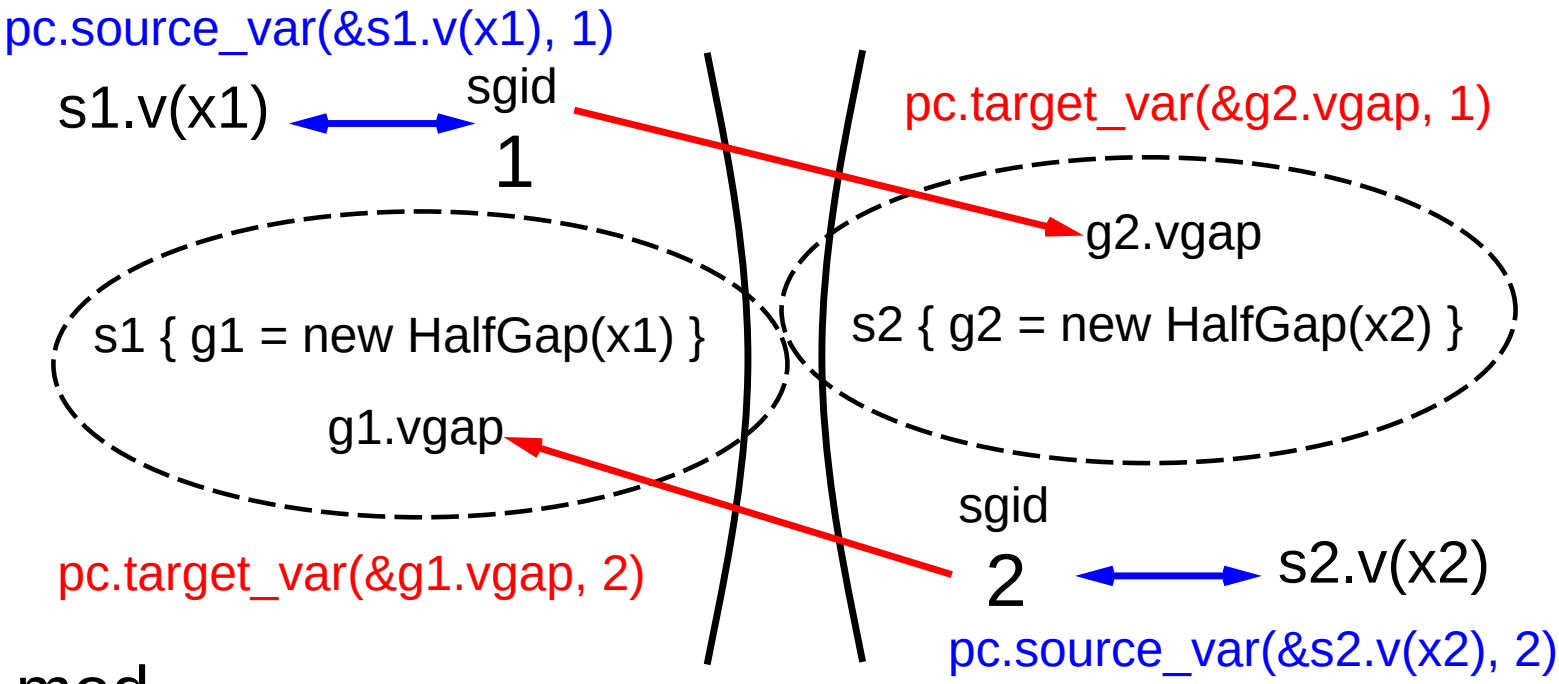
gap.mod

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

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

Continuous Voltage Exchange

pc.source_var(&source_var, sgid)
pc.target_var(&target_var, sgid)



gap.mod

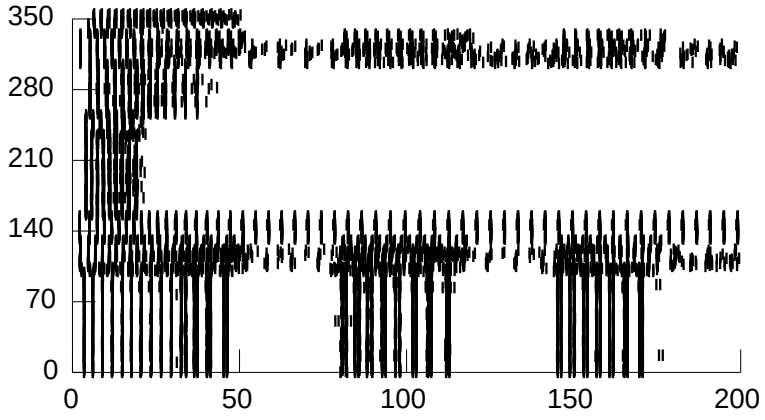
```
NEURON {
  POINT_PROCESS HalfGap
  ELECTRODE_CURRENT i
  RANGE r, i, vgap
}
PARAMETER { r = 1e9 (megohm) }
```

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

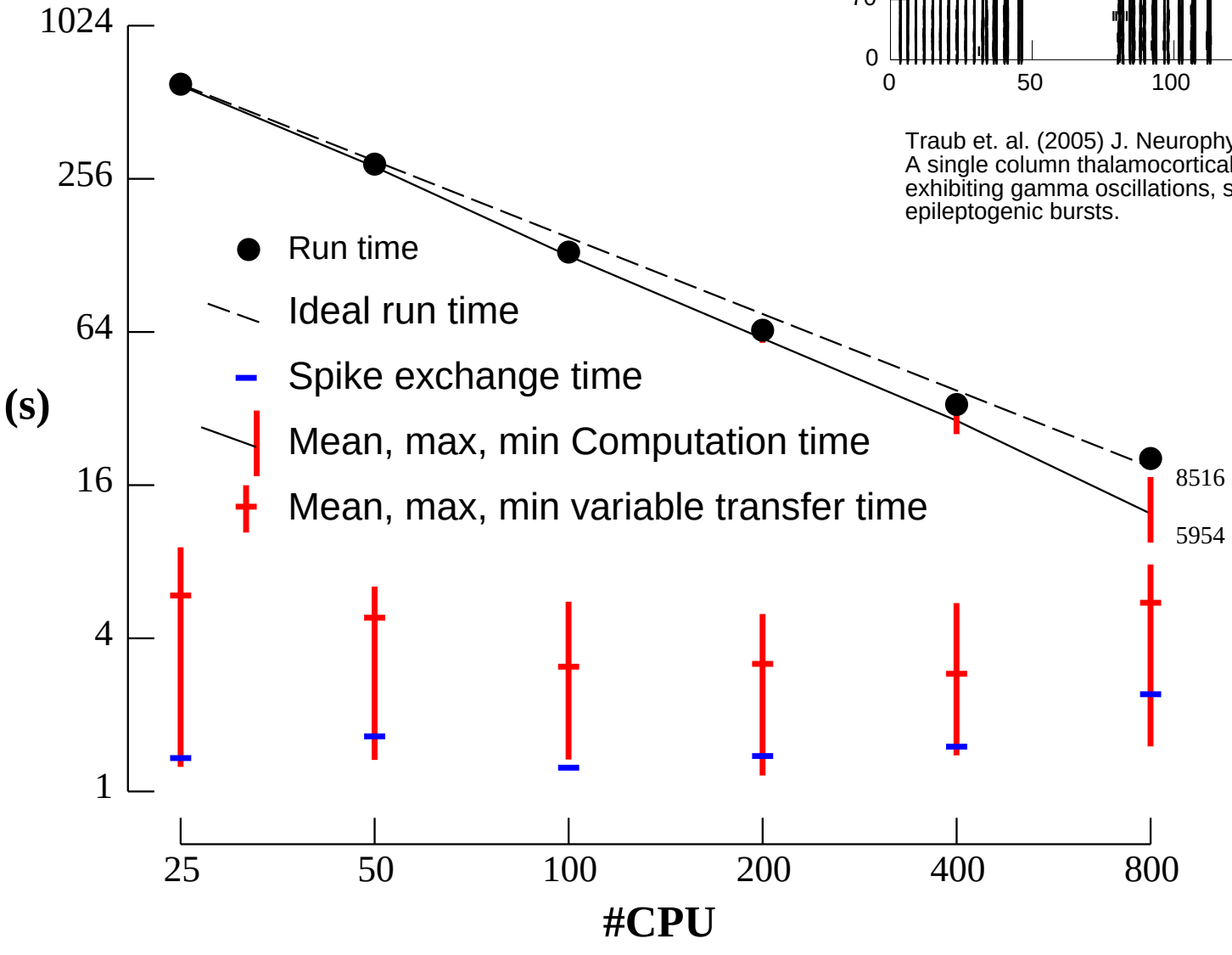
Pittsburgh Supercomputing Center

Bigben Cray XT3

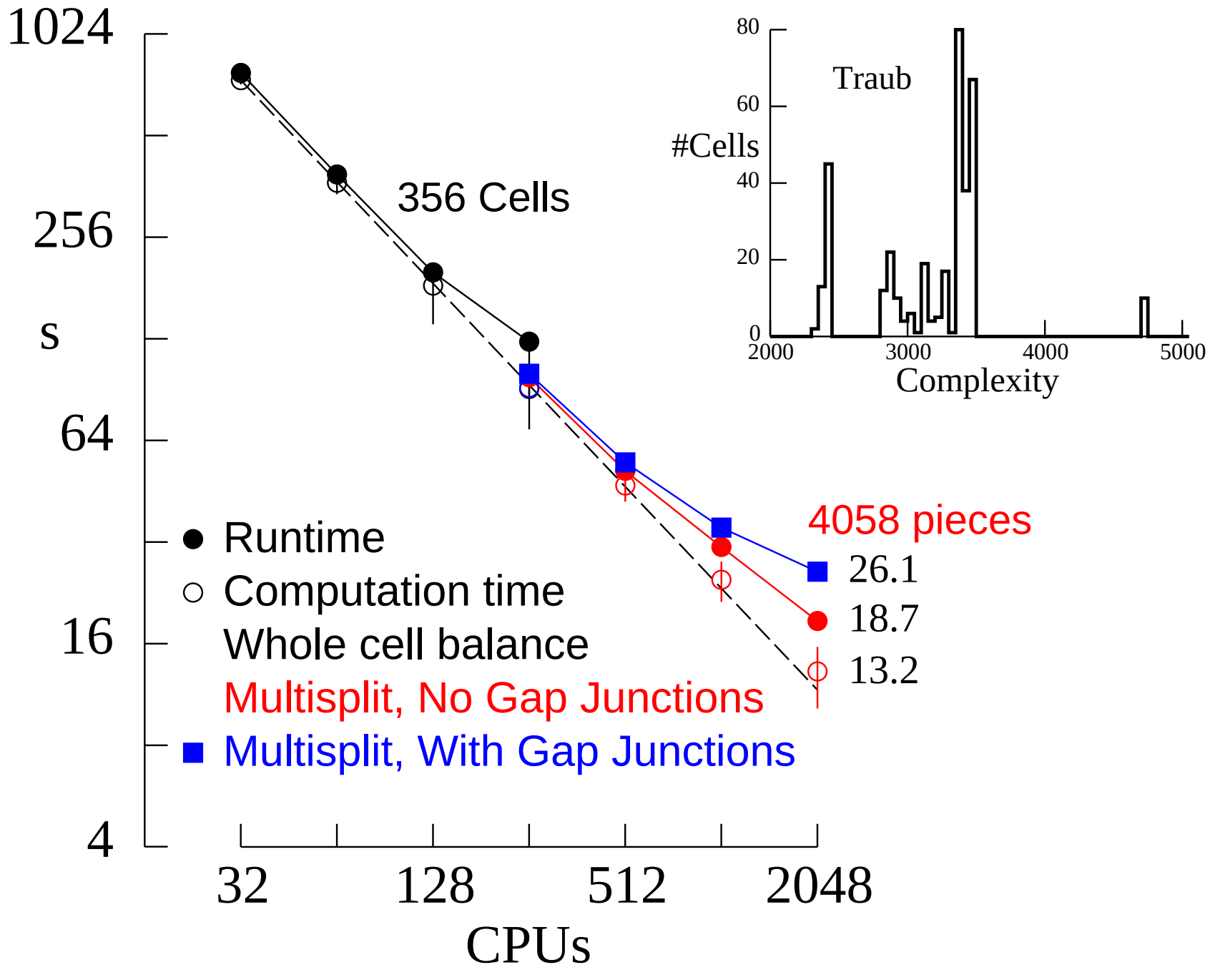
2068 2.4 GHz Opteron Processors



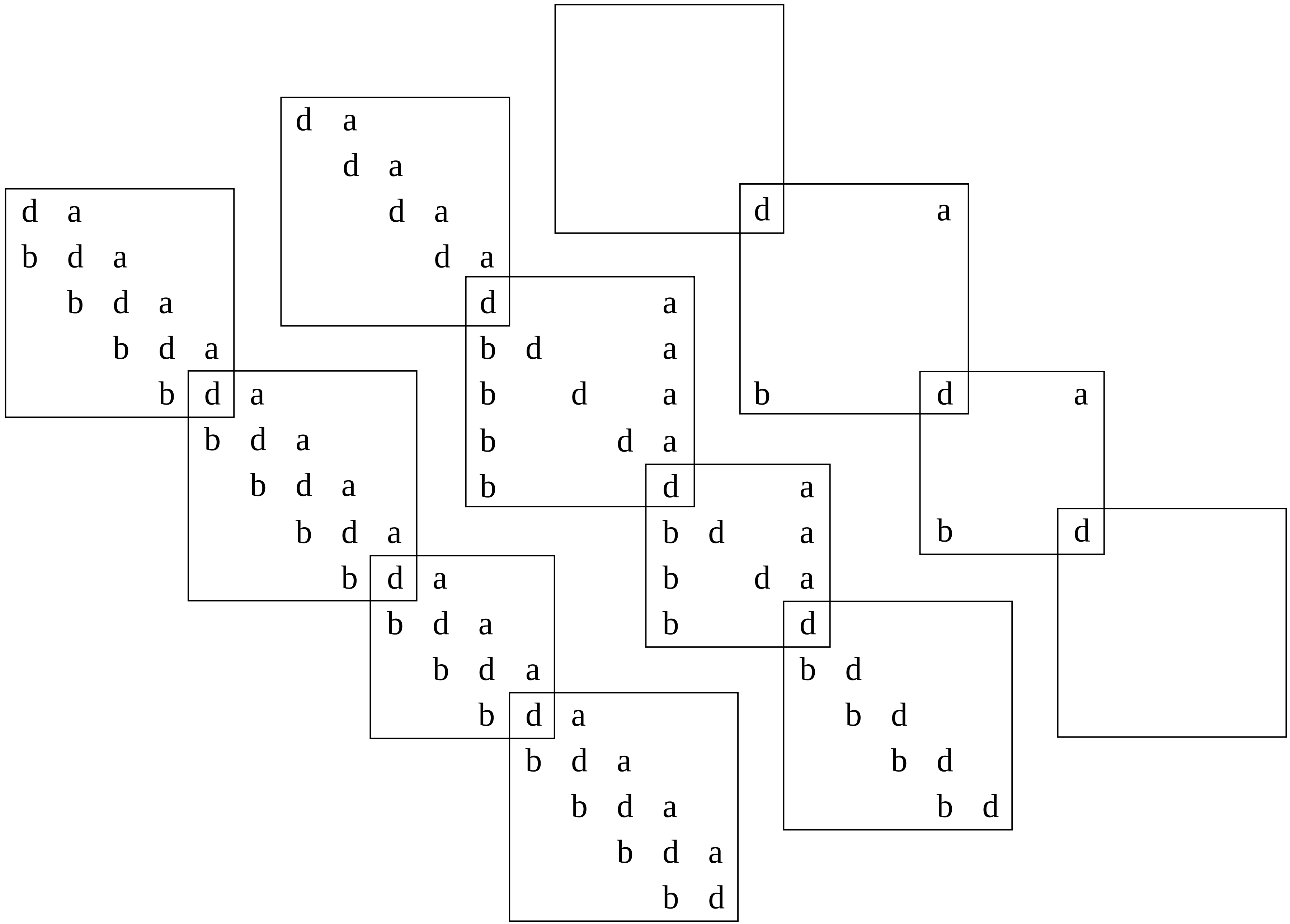
Traub et. al. (2005) J. Neurophysiol 93: 2194
A single column thalamocortical network model exhibiting gamma oscillations, sleep spindles and epileptogenic bursts.

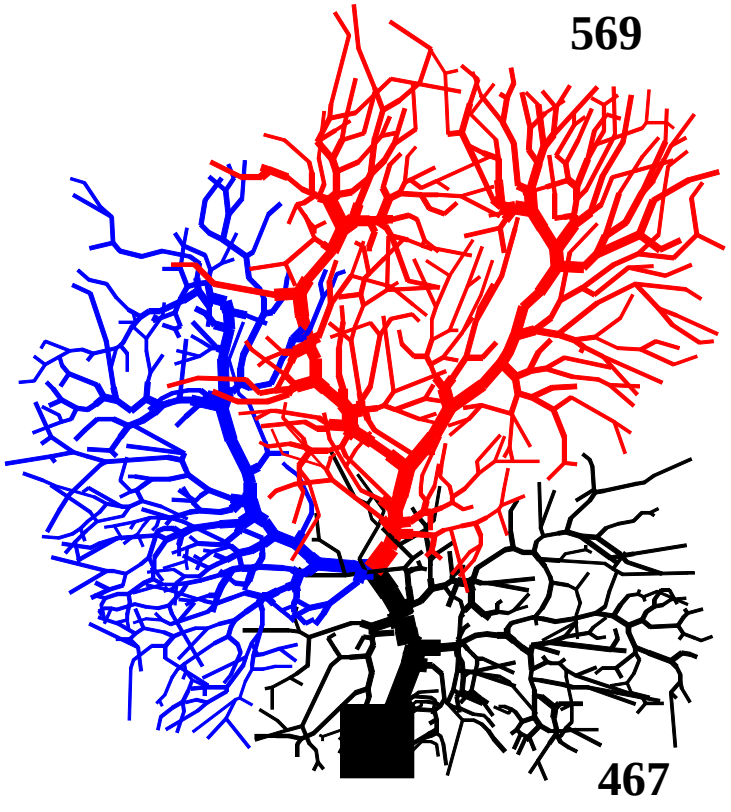


3560 cells 14 types
3500 gap junctions
5,596,810 equations
73,465 spikes
1,122,520 connections
19,844,187 delivered

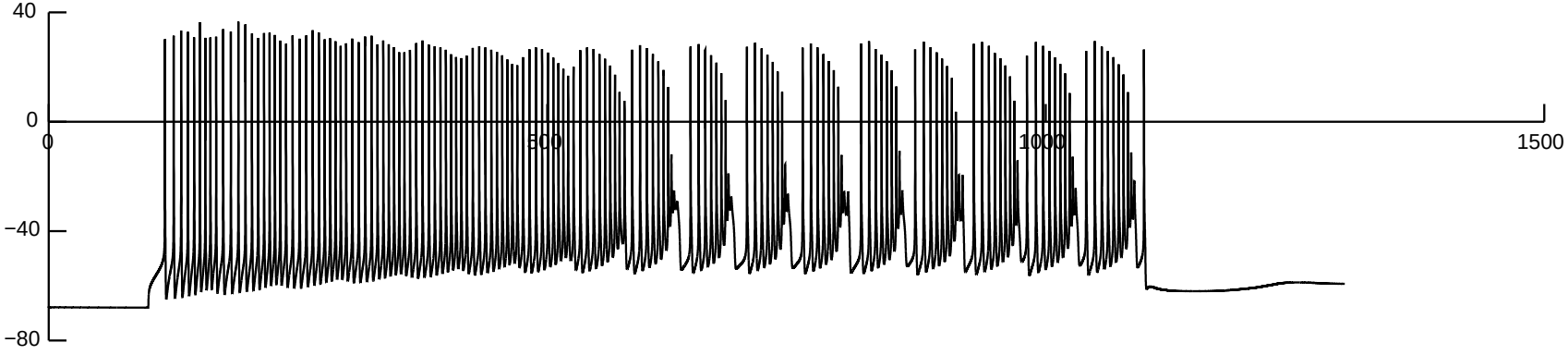

1024
256
64
16
4
s
356 Cells
4058 pieces
26.1
18.7
13.2
Runtime
Computation time
Whole cell balance
Multisplit, No Gap Junctions
Multisplit, With Gap Junctions
32
128
512
2048
CPUs
Traub
#Cells
80
60
40
20
0
2000
3000
4000
5000
Complexity

Multisplit Gaussian Elimination





#CPU	Runtime (s)		
1	54.8		
3	22.2	19.5 expected	18.3 ideal

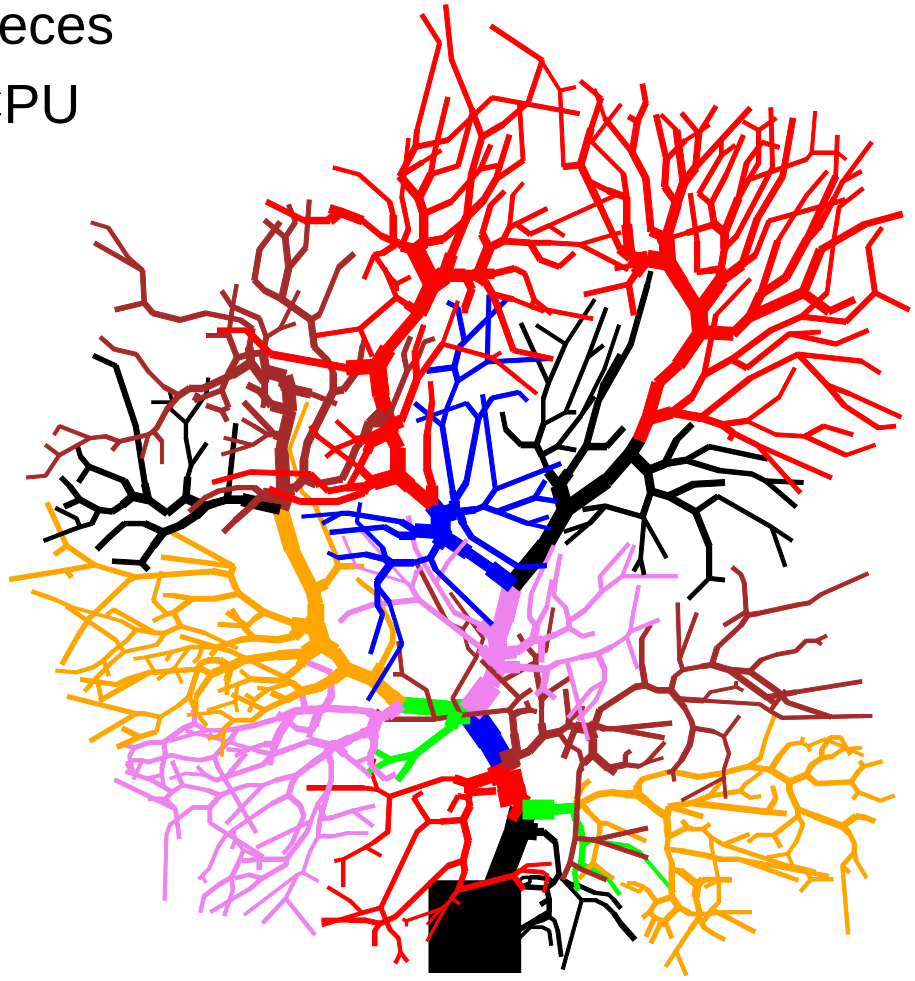


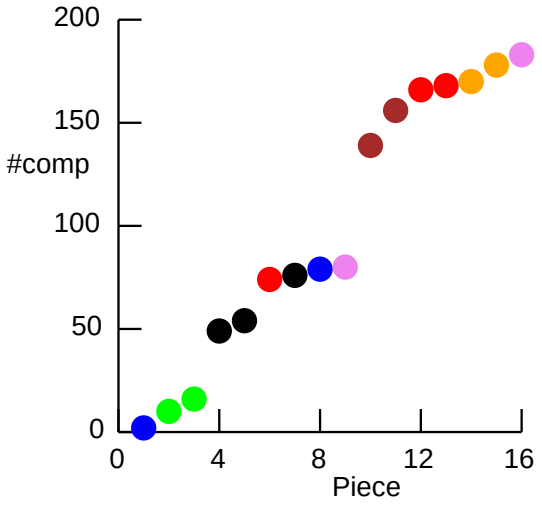
De Schutter & Bower (1994) J. Neurophysiol 71:375

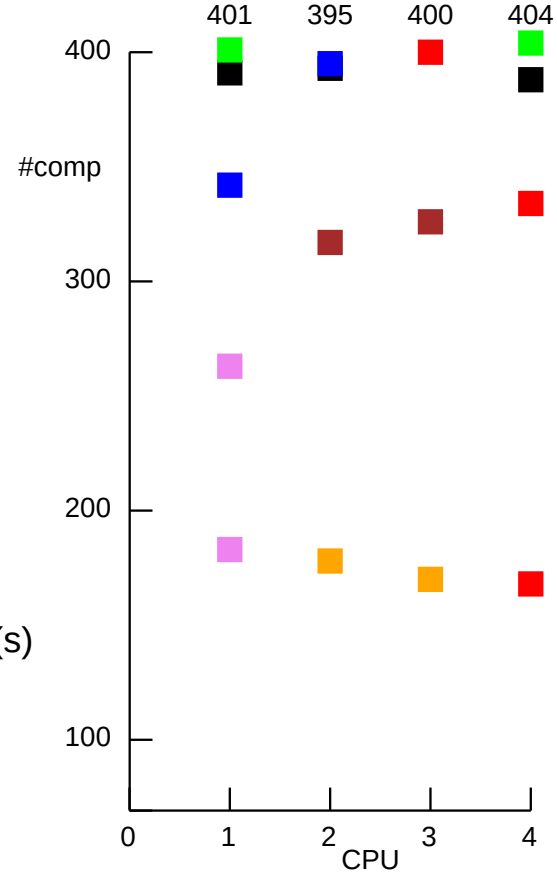
Ported to NEURON by Jenny Davie, Volker Steuber, and Arnd Roth.

16 Pieces

4 CPU



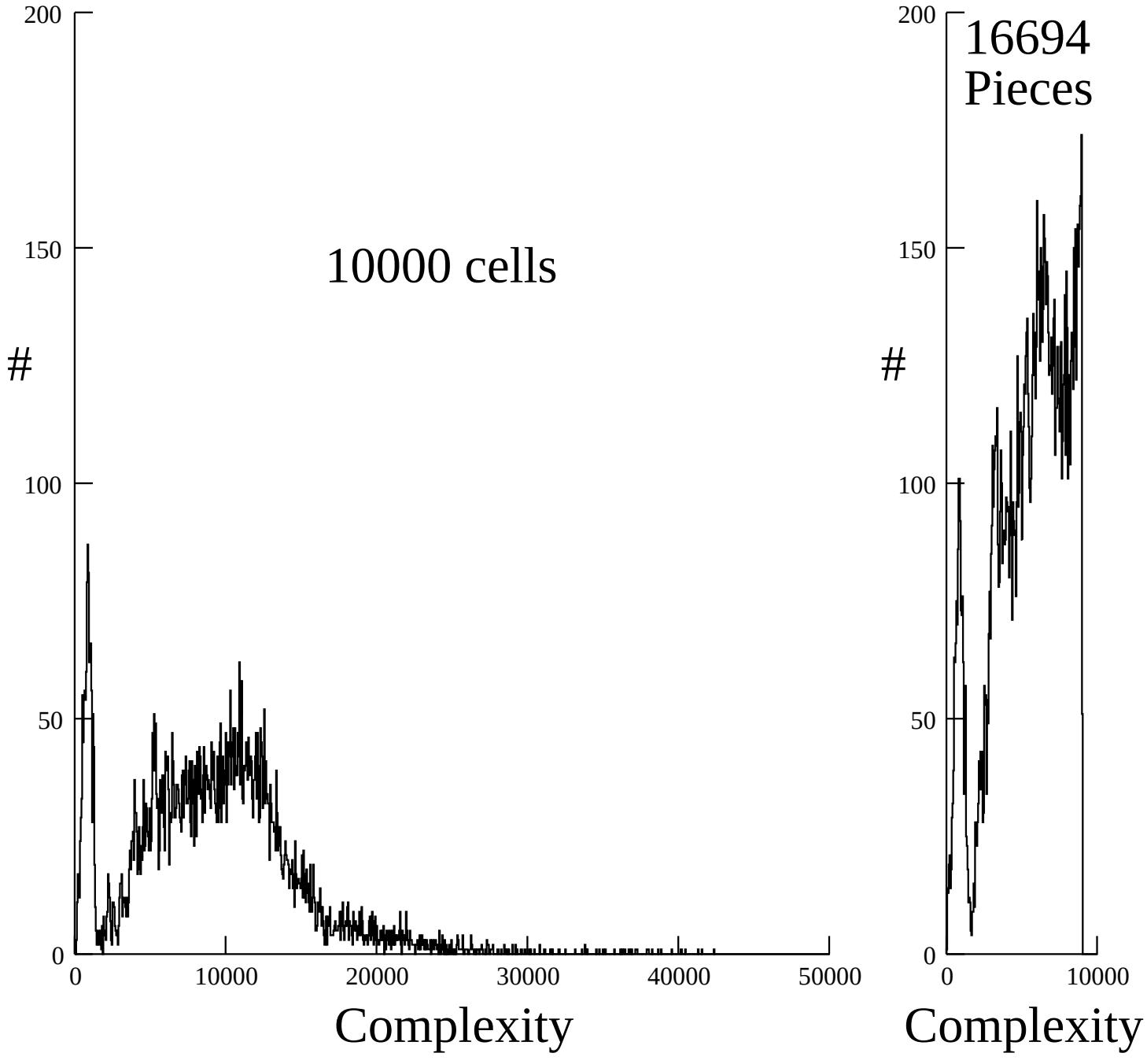


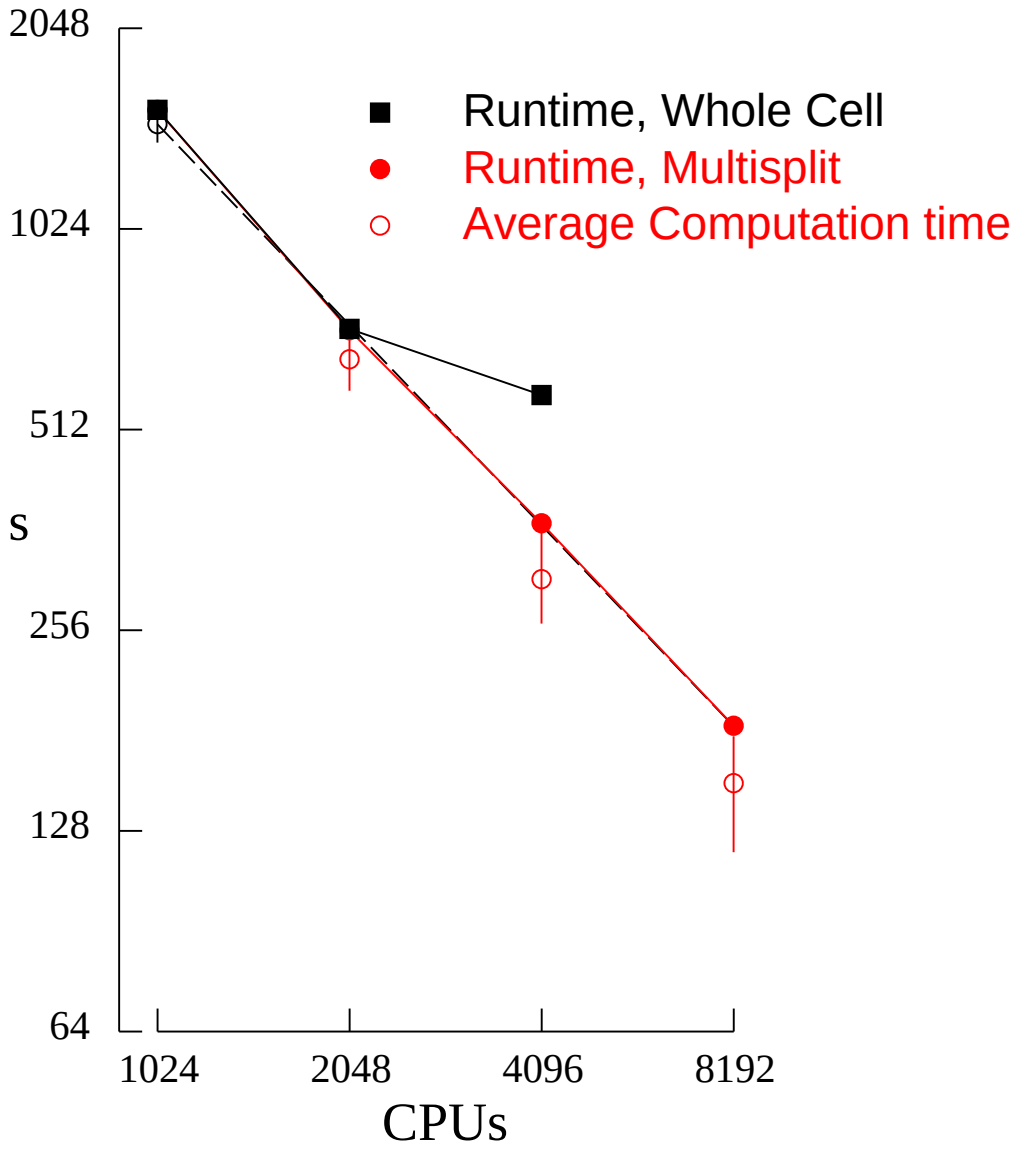


Time (s)

CPU	Computation	Exchange
0	13.82	0.56
1	13.35	1.03
2	13.47	0.90
3	13.56	0.82

	Runtime(s)
16 pieces, 1 cpu	55.0
wholecell, 1 cpu	56.2
16 pieces, 4 cpu	14.4


200
150
100
50
0
#
10000 cells
0
10000
20000
30000
40000
50000
Complexity
200
150
100
50
0
#
16694
Pieces
0
10000
Complexity


2048
1024
512
s
256
128
64
1024
2048
4096
8192
CPUs
Runtime, Whole Cell
Runtime, Multisplit
Average Computation time

Results must be independent of

- Number of processors
- Distribution of cells

Results must be independent of

- Number of processors
- Distribution of cells

What about RANDOM?

- Reproducible
- Independent
- Restartable

Results must be independent of

- Number of processors
- Distribution of cells

What about RANDOM?

- Reproducible
- Independent
- Restartable

Associate a random stream with a cell.

Results must be independent of

- Number of processors
- Distribution of cells

What about RANDOM?

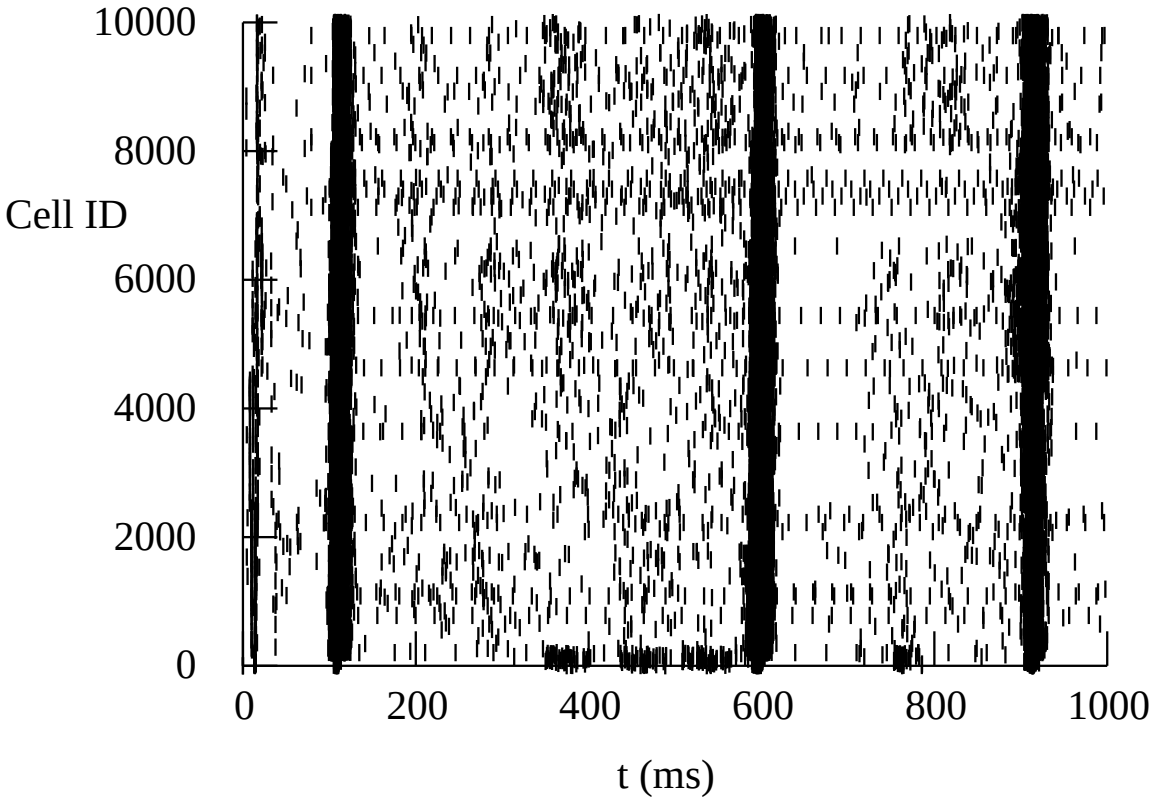
- Reproducible
- Independent
- Restartable

Associate a random stream with a cell.

Use cryptographic transformation of several integers.

- run number
- stream number (cell gid)
- stream pick index

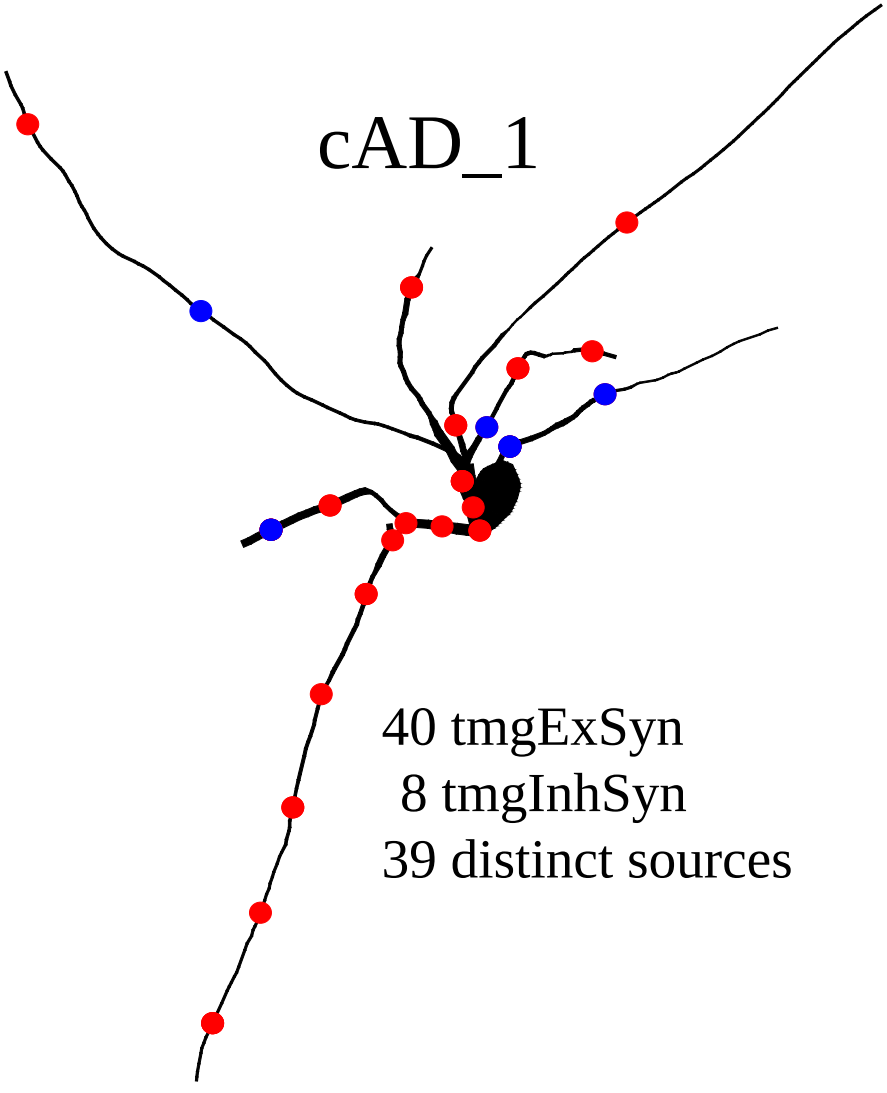
PatternStim

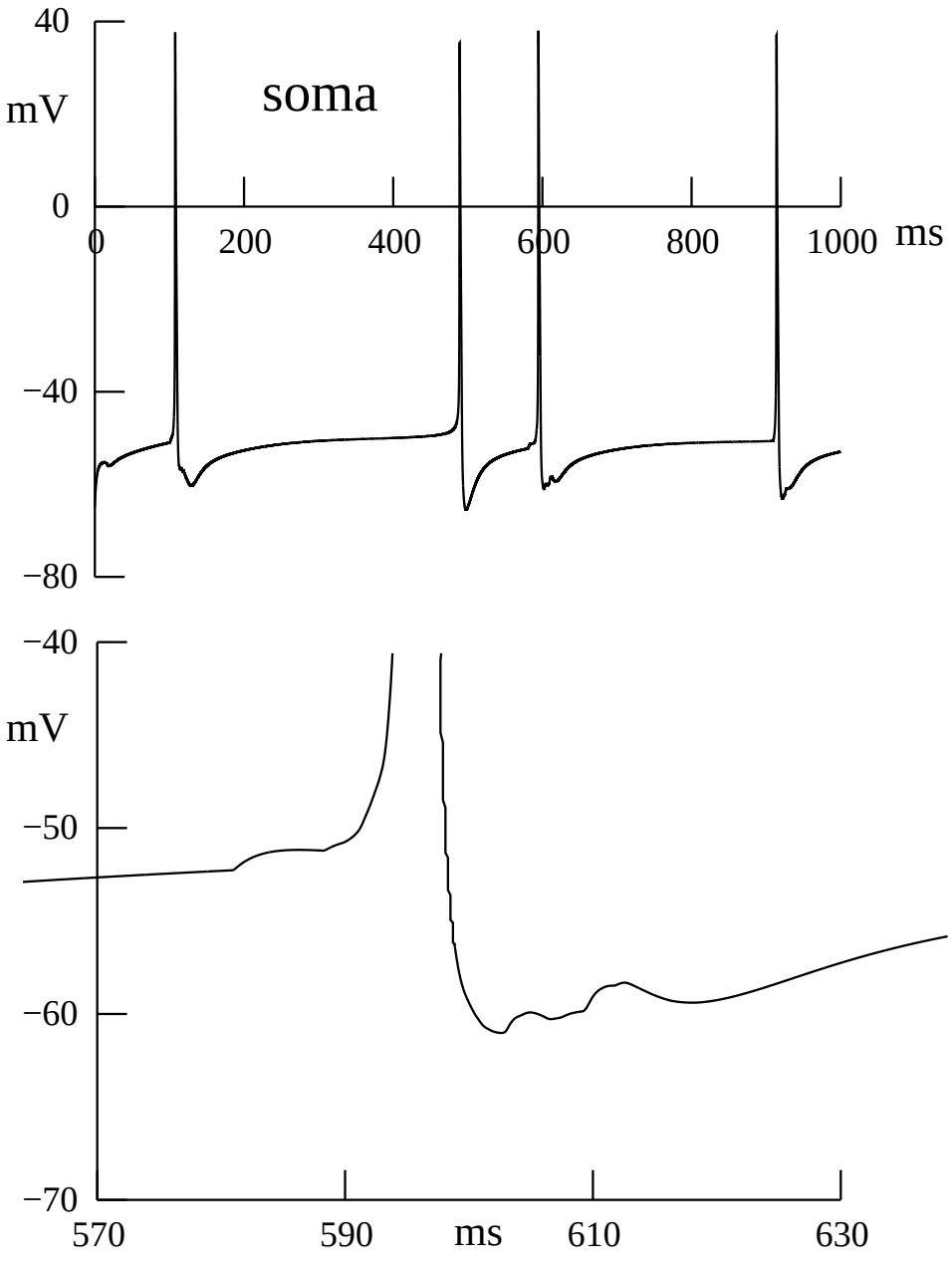


out.spk (58,858 lines)

3.975	8050
3.975	8621
...	
999.125	4632

Line#	spike(ms)	gid
8548:	107.25	1
18501:	488.625	1
27276:	594.25	1
51470:	913.475	1





Debugging

Debugging

1) GID and time of first spike difference.

Debugging

1) GID and time of first spike difference.
2) All spikes delivered to synapses of that Cell?

Debugging

1) GID and time of first spike difference.
2) All spikes delivered to synapses of that Cell?
3) When and what is the first state difference?