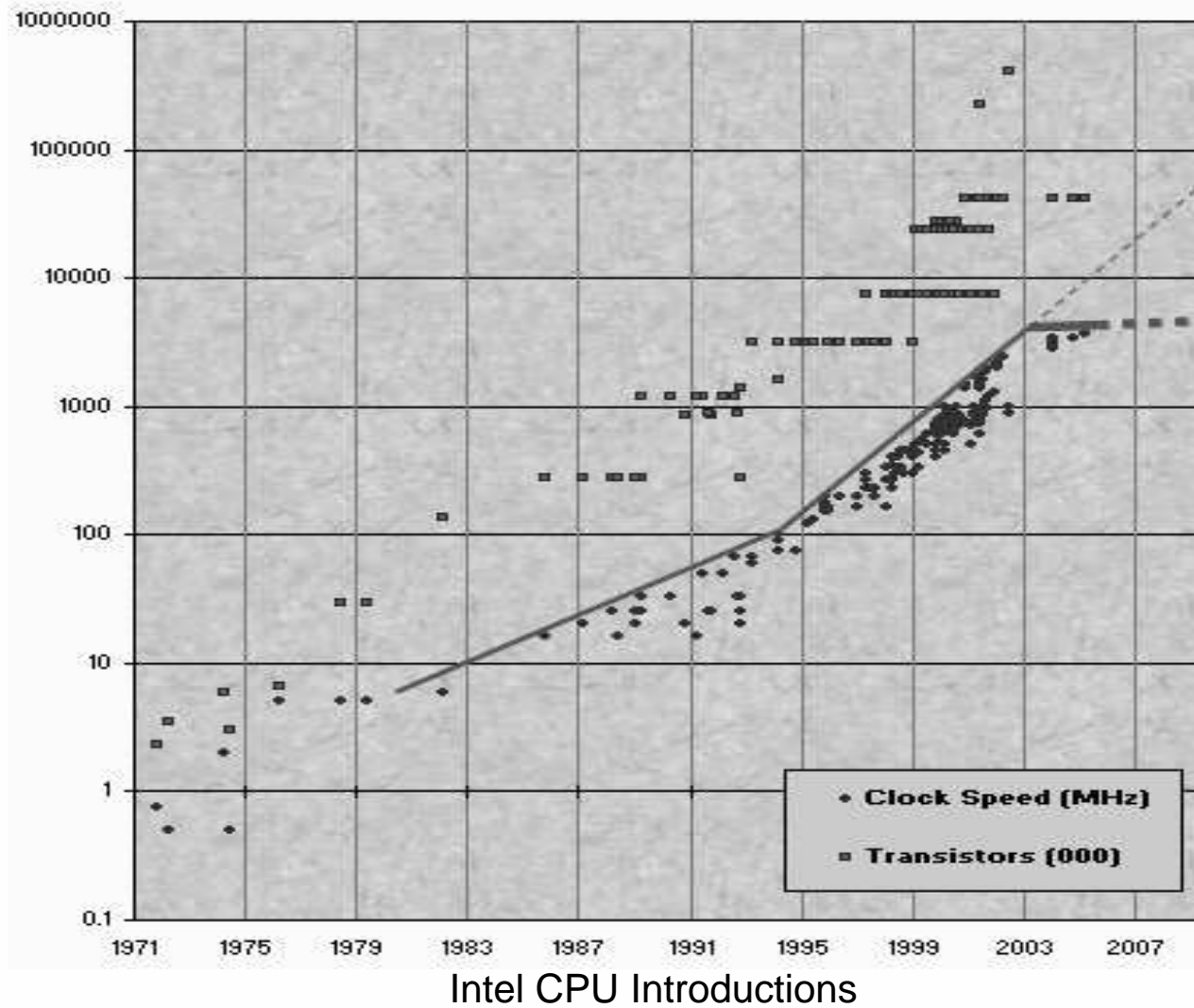


# **NEURON + Threads**

**Simulations on multicore desktops.**

# No (more) Free Lunch



# Thread style in NEURON

## Join

```
run() {  
  while (t < tstop) {  
    multithread_job(step)  
    plot()  
  }  
}
```

```
void* step(NrnThread* nt) {  
  ... nt->id ...  
}
```

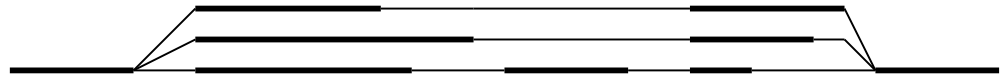


We never use.

## Condition Wait

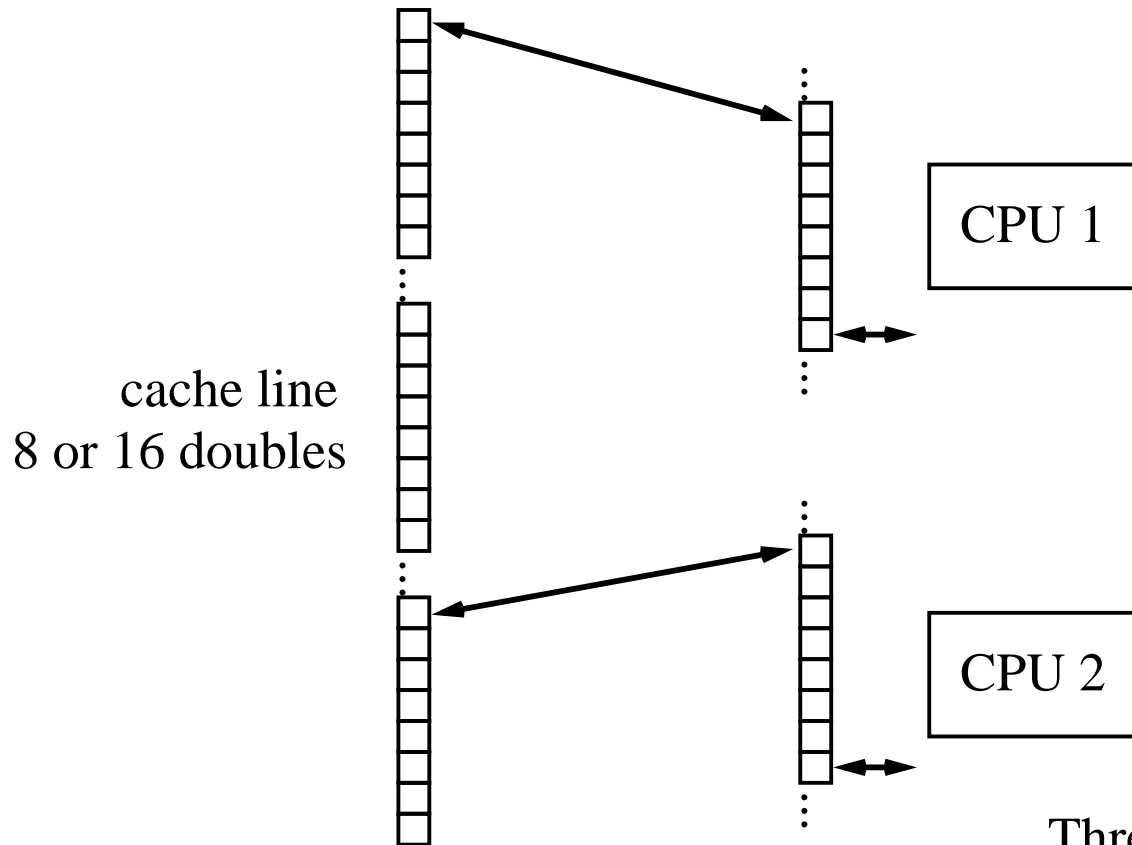
```
multithread_job(run)
```

```
run(NrnThread* nt) {  
  while(t < tstop) {  
    step(nt)  
    barrier()  
    if (nt->id == 0) { plot() }  
    barrier()  
  }  
}
```



Reminiscent of MPI

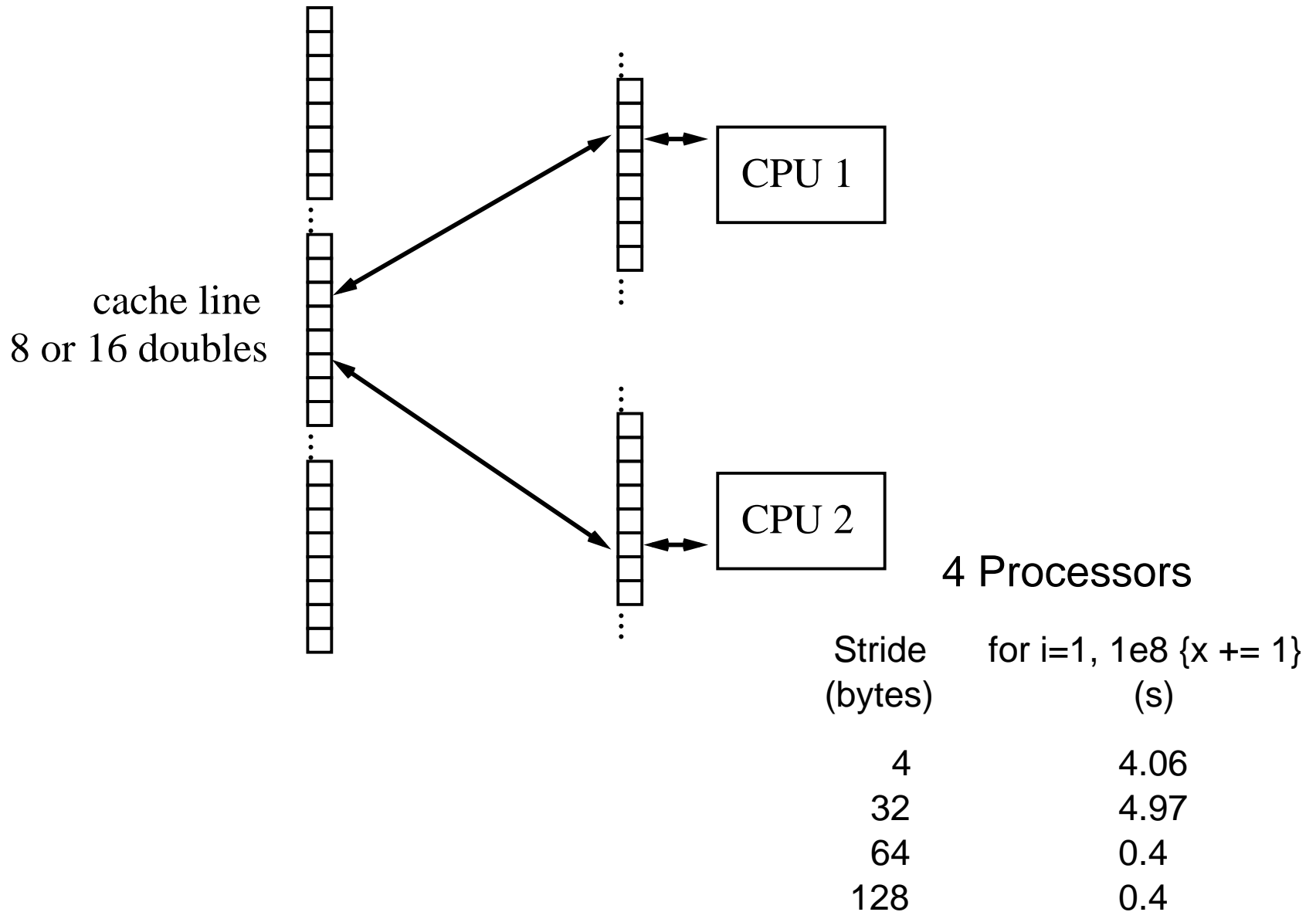
# Ideal cache efficiency



10000 passive compartments  
4 core 3GHz x86\_64

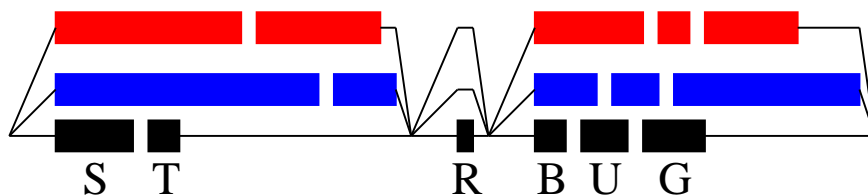
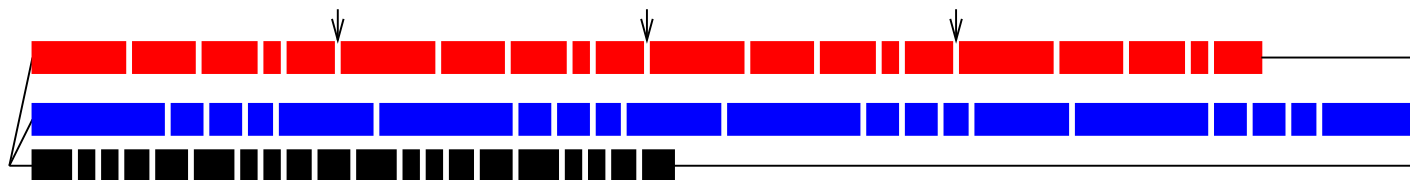
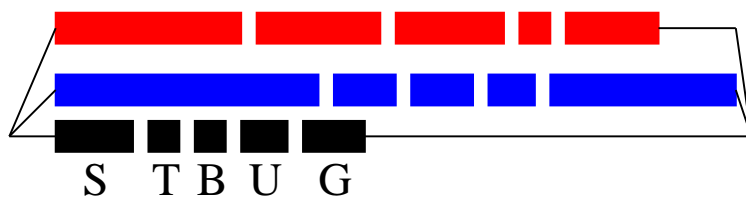
Threads	Runtime (s)	
	Cache Off	Cache On
1	4.92	0.45
2	1.14	0.23
4	0.29	0.12
8	0.23	0.09

# False cache line sharing

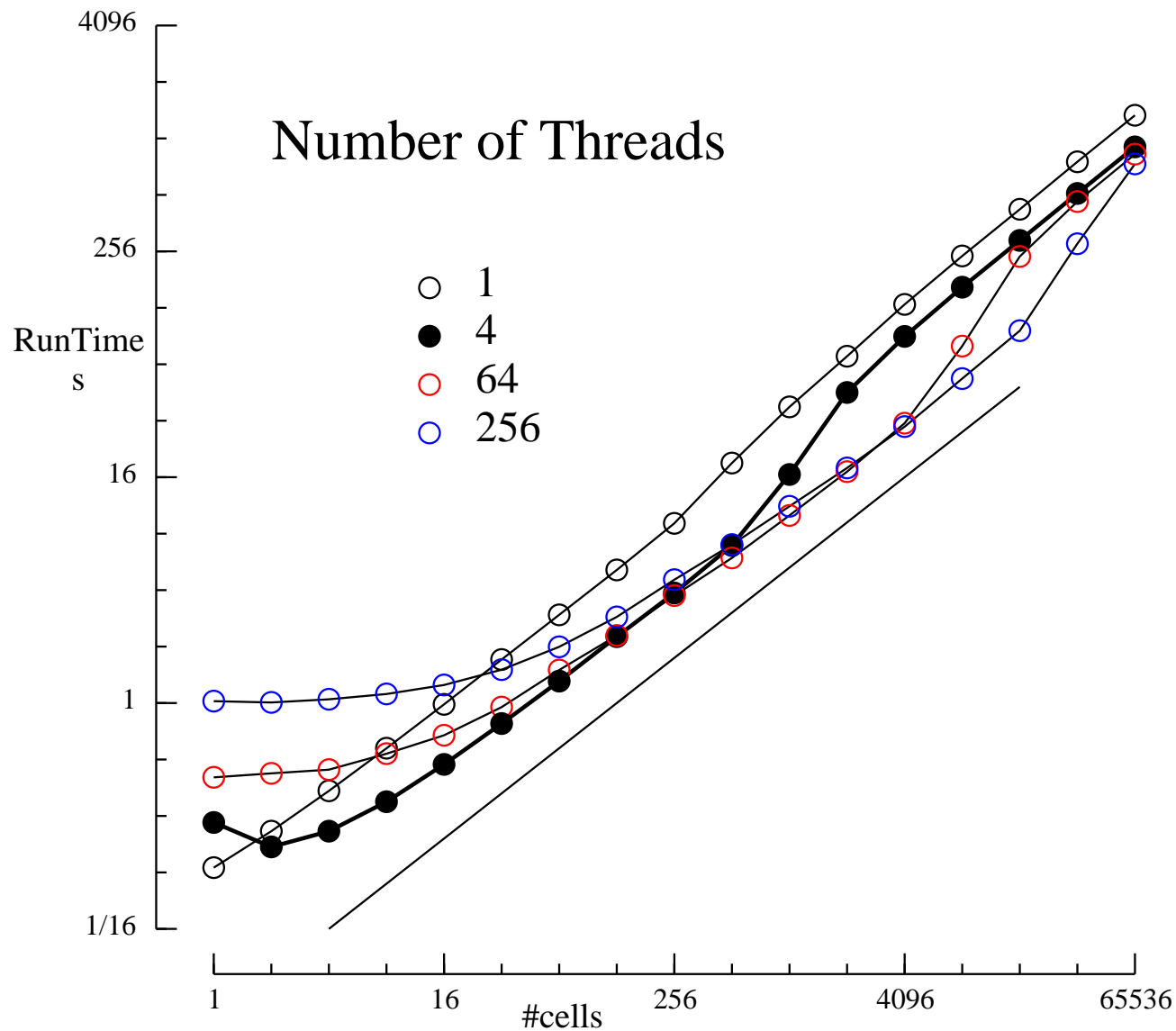


Fixed step:  $t \rightarrow t + dt$

S	T	R	B	U	G
setup	triang	reduce solve	bksub	update	cond gates

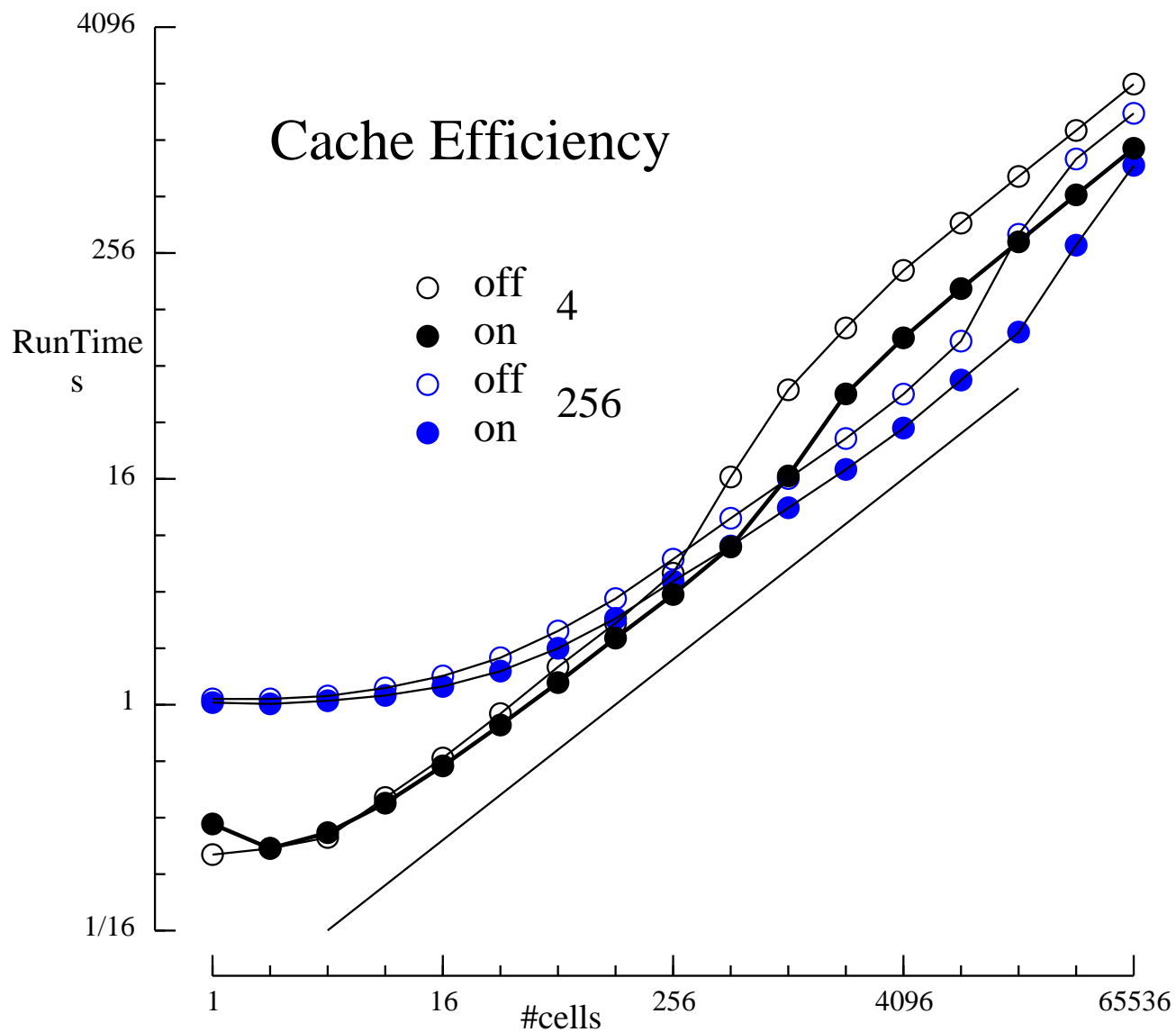


Global var  $dt$   $y' = f()$   $dy'/dy$   
 27 ||Vector operations



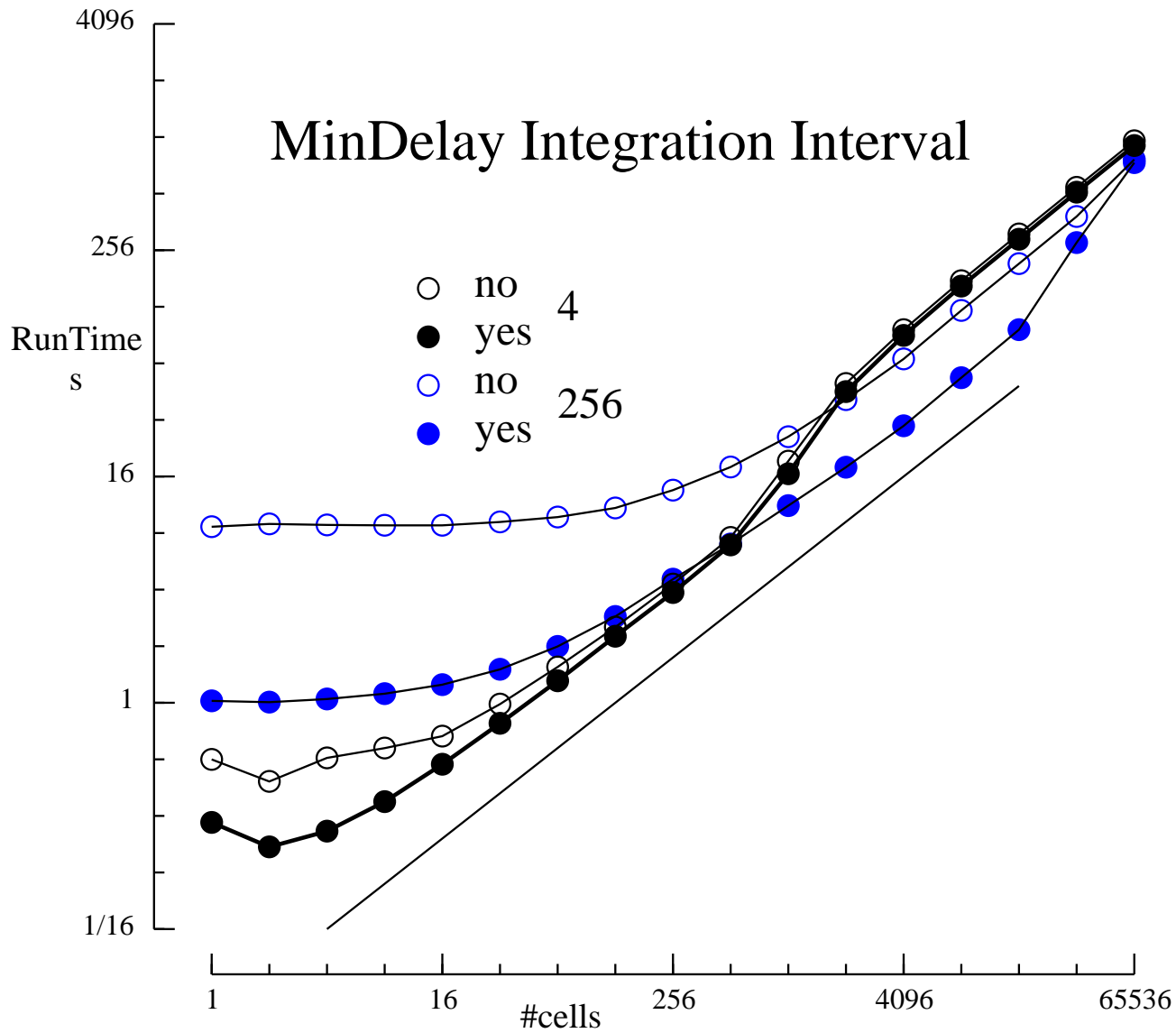
cell: 11 HH cmpts

4-core machine



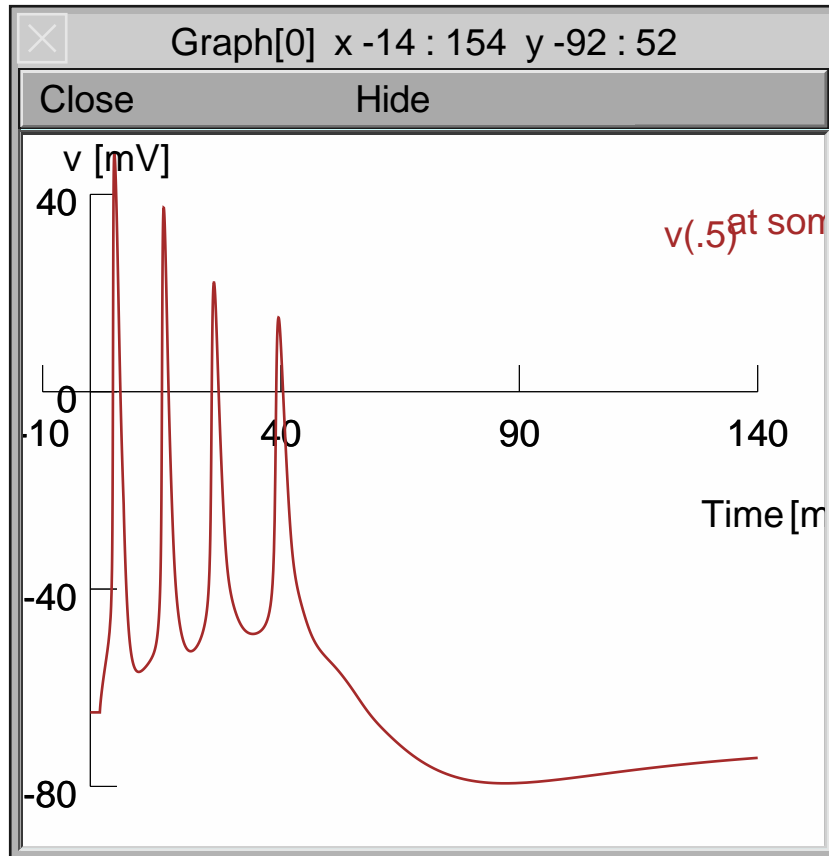
4-core machine





4-core machine

# Lazarewicz 2002, CA3 Pyramidal Neuron



RunControl

Close Hide

Init (mV) ← -65

Init & Run

Stop

Continue til (ms) ← 5

Continue for (ms) ← 1

Single Step

t (ms) 140

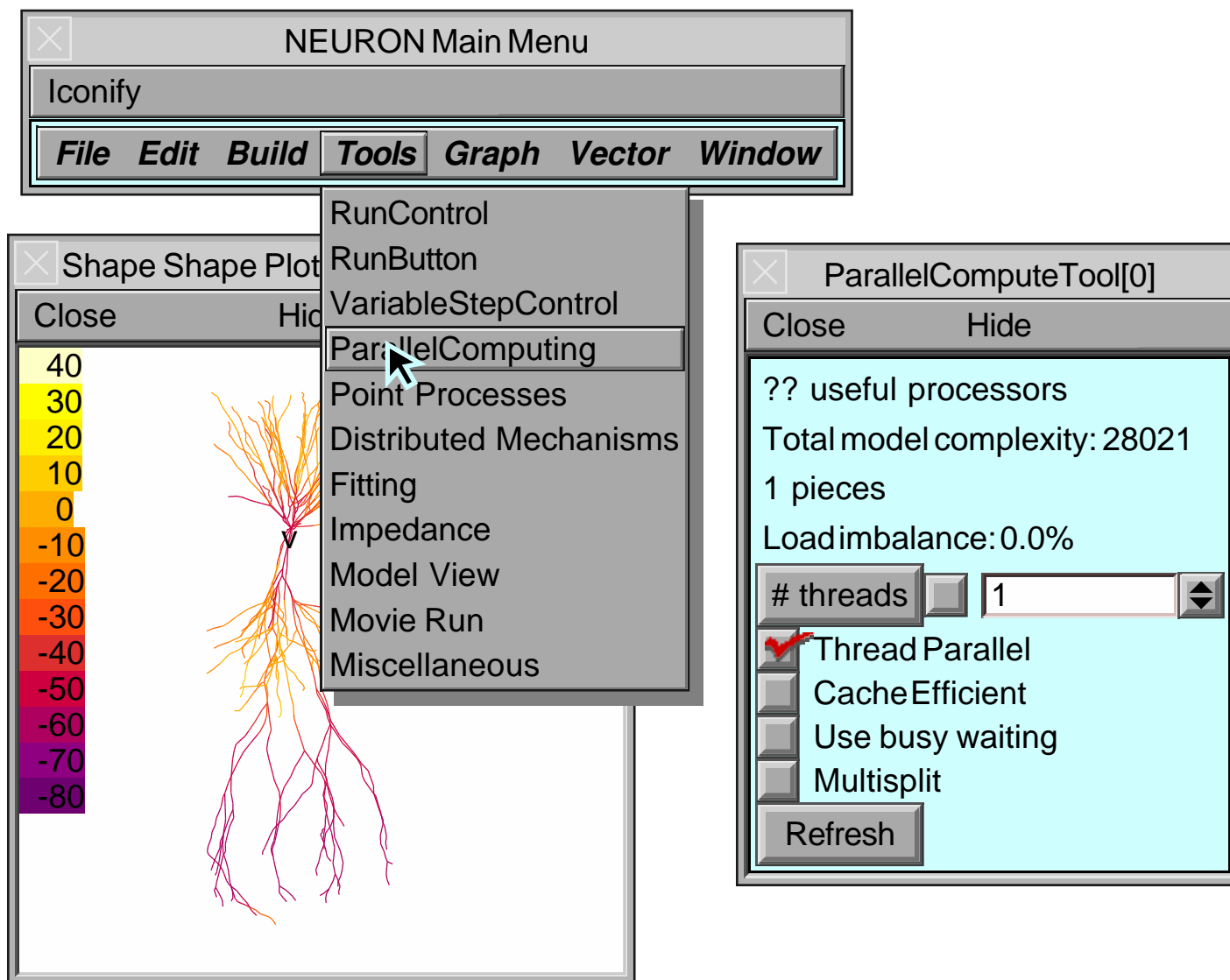
Tstop (ms) 140

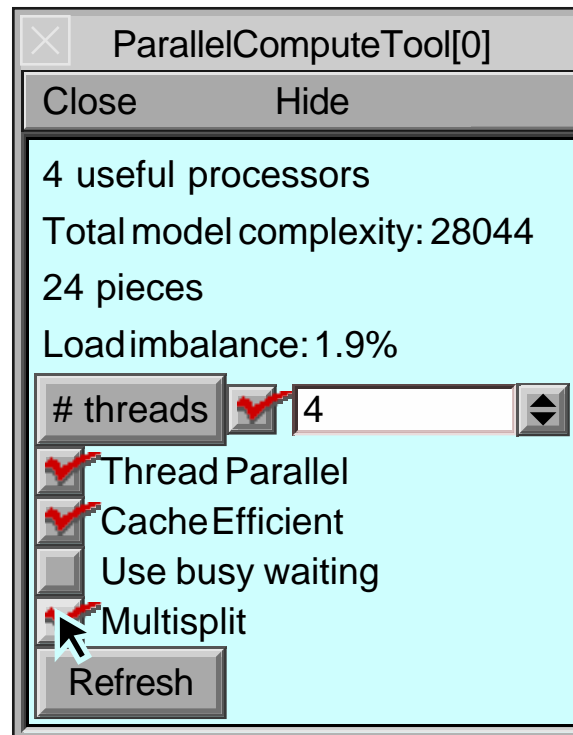
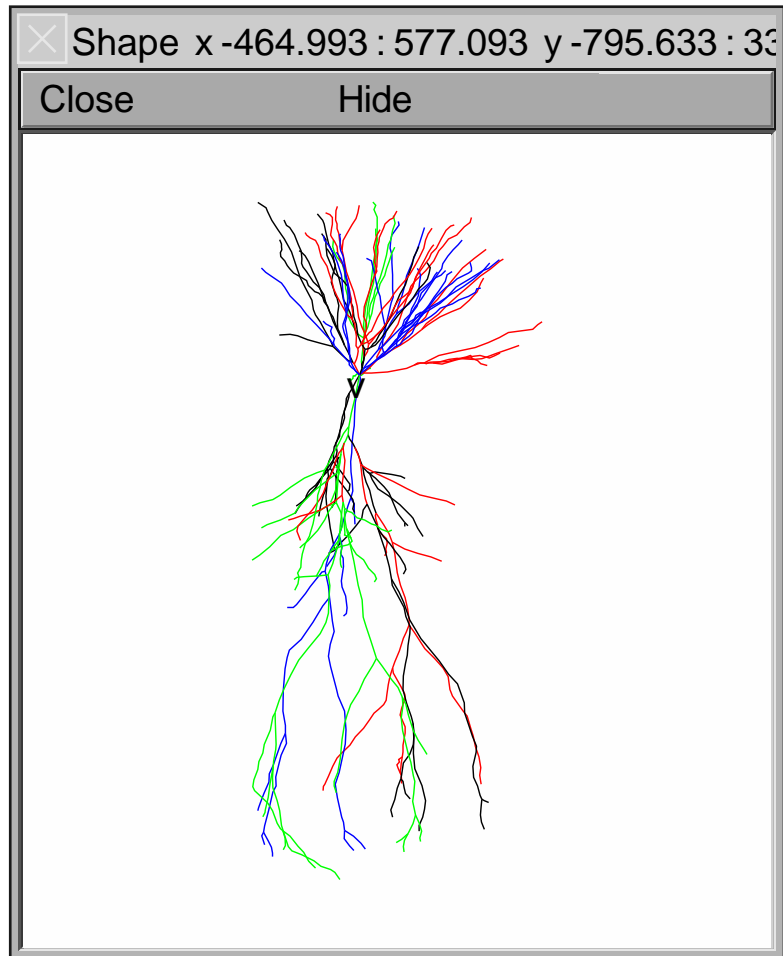
dt (ms) ☒ 2.335

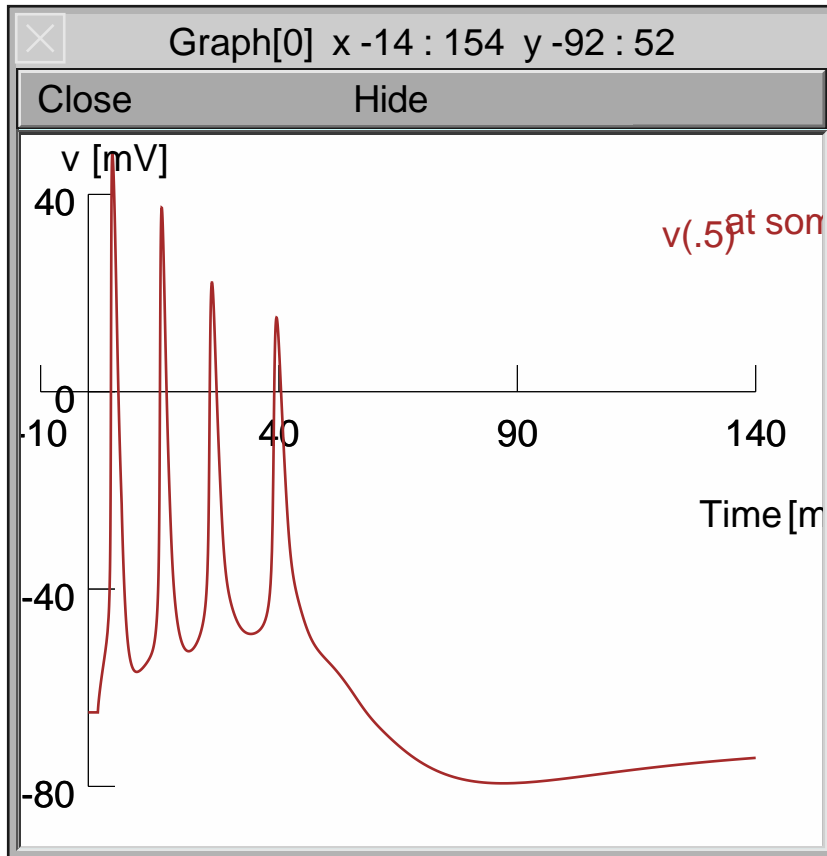
Points plotted/ms 40

Scrn update invl (s) 0.05

Real Time (s) 35.4







RunControl

Close Hide

Init (mV) ← -65

Init & Run

Stop

Continue til (ms) ← 5

Continue for (ms) ← 1

Single Step

t (ms) 140

Tstop (ms) 140

dt (ms) ☒ 2.3008

Points plotted/ms 40

Scrn update invl (s) 0.05

Real Time (s) 9.77

instead of 35.4s

**\$ mkthreadsafe**

NEURON {

SUFFIX CA1M95

USEION ca READ cai,cao WRITE ica

RANGE gbar,ica

GLOBAL minf,tau

}

Translating CA1M95.mod into CA1M95.c

Notice: Assignment to the GLOBAL variable, "minf", is not thread safe

Notice: Assignment to the GLOBAL variable, "tau", is not thread safe

Force THREADSAFE? [y][n]: n

```
DERIVATIVE state {  
    rate(v)  
    m' = (minf - m)/tau  
}
```

```
PROCEDURE rate(v (mV)) {  
    LOCAL a  
    a = alp(v)  
    tau = 1/(tfa*(a + bet(v)))  
    minf = tfa*a*tau  
}
```

Force THREADSAFE? [y][n]: n  
y

```
NEURON {  
  THREADSAFE  
  SUFFIX CA1M95  
  USEION ca READ cai,cao WRITE ica  
  RANGE gbar,ica  
  GLOBAL minf,tau  
}
```



**\$ mkthreadsafe**

```
NEURON {  
    POINT_PROCESS GABAa  
    POINTER pre  
    ...  
}  
  
    VERBATIM  
    return 0;  
    ENDVERBATIM
```

Translating gabaa.mod into gabaa.c

Notice: Use of POINTER is not thread safe.

Notice: VERBATIM blocks are not thread safe

Notice: Assignment to the GLOBAL variable, "Rtau", is not thread safe

Notice: Assignment to the GLOBAL variable, "Rinf", is not thread safe

Force THREADSAFE? [y][n]: n

# Relative thread performance

## 1 vs 4 processors

