



# Scalable Performance Analysis Using Statistical Clustering

---

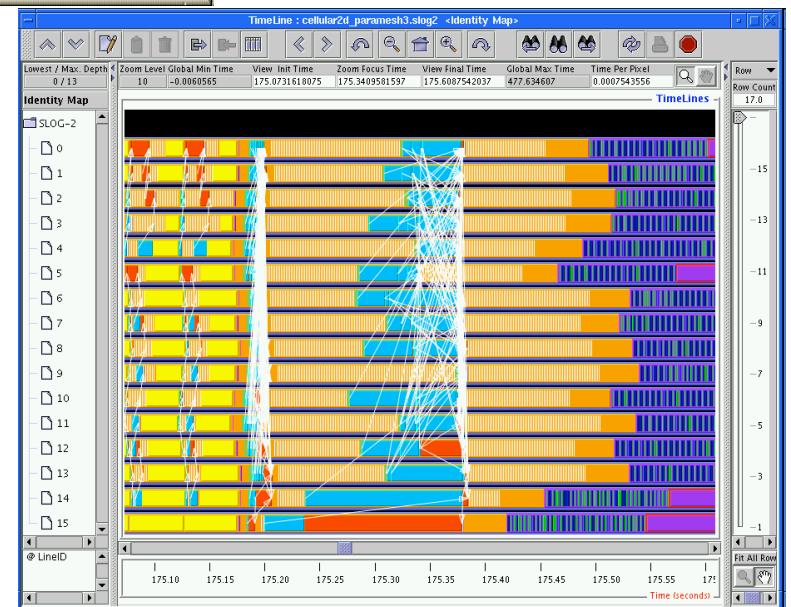
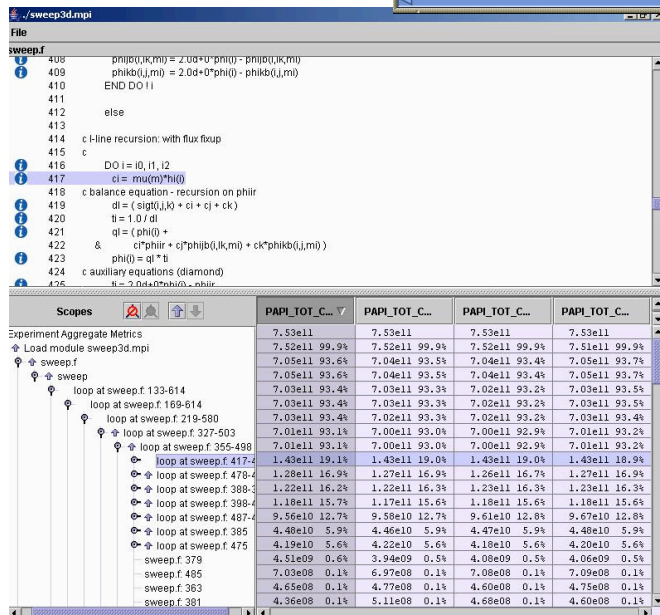
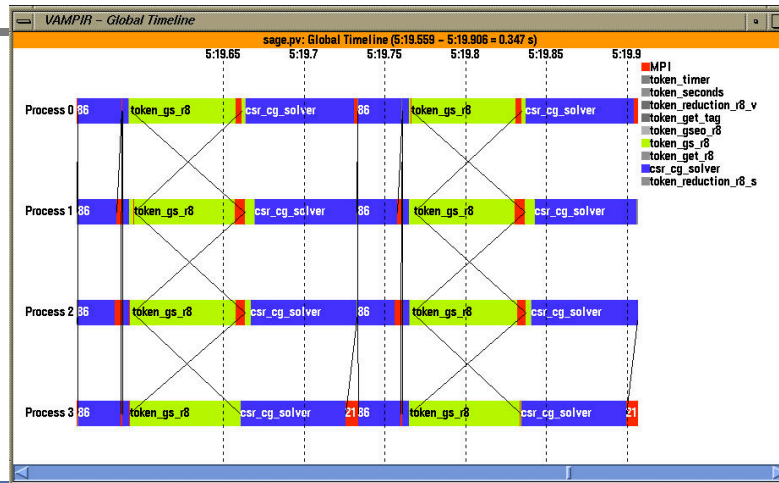
Adam Bordelon, Rob Fowler,  
Bradley Broom, and Ken Kennedy  
Department of Computer Science  
Rice University



# Supercomputers need Super Tools

- Parallel systems and applications are constantly increasing in scale and complexity
- We can profile these systems with very low overhead and generate detailed performance data
- What can we do with all this data to make it useful to the human analyst?

# Current Tools Don't Scale





# Scalable Data Management

---

- **Problem 1:** Find performance anomalies without bogging down a single node
- Assume numerous performance profiles each stored locally on its compute node
- Strategy:
  - Take advantage of the inherent similarities in performance between individual nodes.
  - Exploit the resources of the parallel system to scalably manage data aggregation and comparison.



# Approach

---

- 1. Sample Profiles:** Sample a random subset of nodes and retrieve full performance profiles from these nodes.
- 2. Generate Summary:** Using these full profiles, estimate the average system profile. Based on user or performance-defined bounds, generate an application performance summary with detailed performance data in the code regions of interest, and only higher-level information for less important regions.
- 3. Distributed Diff:** Broadcast the performance summary to the unsampled nodes. They locally compare their performance against the summary and return key differences to the user-side tool.



# Scalable Data Analysis

---

- **Problem 2:** Finding complex performance patterns in large-scale data
- HPCToolkit performance data output
  - Hierarchical format with aggregate measures at each level of source construct
  - Measured performance counter values listed for each metric on each node at each level
- What do we want to find?
  - Application performance characteristics
  - System variation across nodes
  - Anomalies (app bugs, system effects)



# Statistical Clustering

---

- Similarity measure defines clustering
- 1-way: k-means, hierarchical, gene-shaving, etc
- 2-way: biclustering
  - constant values, coherent values or biclusters with coherent evolutions
  - structures: single biclusters, overlapping or not overlapping biclusters, etc

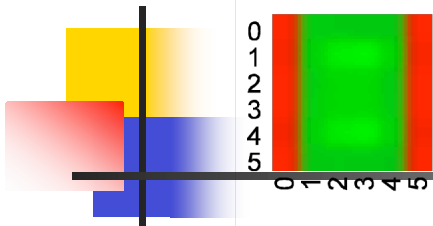


# Performance Data Analysis

---

- Transform the data to 2D matrix
  - Rows: Source code regions
  - Columns: Processors (or threads)
  - Values: Performance metrics
- Run the performance matrix through one of the clustering algorithms
- Output should give us:
  - Subspace clusters: Groups of code regions that behave similarly on sets of nodes
  - Node partitions: high/low performance

# Cluster 1: 62% of variance in Sweep3D



Weight	Clone ID
-6.39088	sweep.f,sweep:260
-7.43749	sweep.f,sweep:432
-7.88323	sweep.f,sweep:435
-7.97361	sweep.f,sweep:438
-8.03567	sweep.f,sweep:437
-8.46543	sweep.f,sweep:543
-10.08360	sweep.f,sweep:538
-10.11630	sweep.f,sweep:242
-12.53010	sweep.f,sweep:536
-13.15990	sweep.f,sweep:243
-15.10340	sweep.f,sweep:537
-17.26090	sweep.f,sweep:535

```

c I-inflows for block (i=i0 boundary)
c
  if (ew_rcv .ne. 0) then
    call rcv_real(ew_rcv, phiib, nib, ew_tag, info)
  else
    if (i2.lt.0 .or. ibc.eq.0) then
      do mi = 1, mmi
      do lk = 1, nk
      do j = 1, jt
        phiib(j,lk,mi) = 0.0d+0
      end do
      end do
      end do
    
```

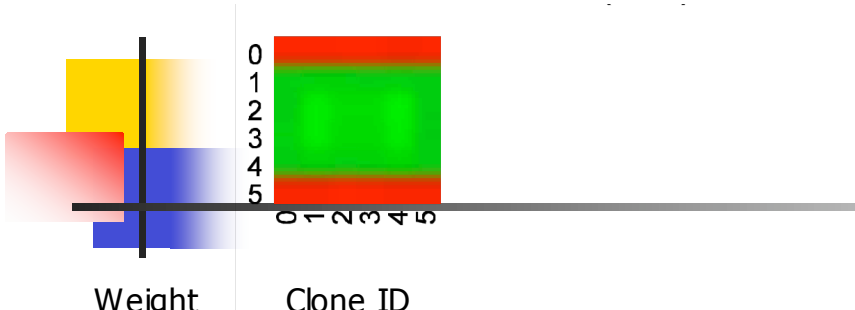
```

  if (ew_snd .ne. 0) then
    call snd_real(ew_snd, phiib, nib, ew_tag, info)
  c   nmess = nmess + 1
  c   mess = mess + nib
  else
    if (i2.lt.0 .and. ibc.ne.0) then
      leak = 0.0
      do mi = 1, mmi
        m = mi + mio
      do lk = 1, nk
        k = k0 + sign(lk-1,k2)
      do j = 1, jt
        phiibc(j,k,m,k3,j3) = phiib(j,lk,mi)
        leak = leak
&          + wmu(m)*phiib(j,lk,mi)*dj(j)*dk(k)
      end do
      end do
      end do
      leakage(1+i3) = leakage(1+i3) + leak
    else
      leak = 0.0
      do mi = 1, mmi
        m = mi + mio
      do lk = 1, nk
        k = k0 + sign(lk-1,k2)
      do j = 1, jt
        leak =leak+ wmu(m)*phiib(j,lk,mi)*dj(j)*dk(k)
      end do
      end do
      end do
      leakage(1+i3) = leakage(1+i3) + leak
    endif
  endif
end if
end if

```



# Cluster 2: 36% of variance



Weight	Clone ID
-6.31558	sweep.f,sweep:580
-7.68893	sweep.f,sweep:447
-7.79114	sweep.f,sweep:445
-7.91192	sweep.f,sweep:449
-8.04818	sweep.f,sweep:573
-10.45910	sweep.f,sweep:284
-10.74500	sweep.f,sweep:285
-12.49870	sweep.f,sweep:572
-13.55950	sweep.f,sweep:575
-13.66430	sweep.f,sweep:286
-14.79200	sweep.f,sweep:574

```

c J-inflows for block (j=j0 boundary)
c
  if (ns_rcv .ne. 0) then
    call rcv_real(ns_rcv, phijb, njb, ns_tag, info)
  else
    if (j2.lt.0 .or. jbc.eq.0) then
      do mi = 1, mmi
        do lk = 1, nk
          do i = 1, it
            phijb(i,lk,mi) = 0.0d+0
          end do
        end do
      end do
    end do
  end do

```

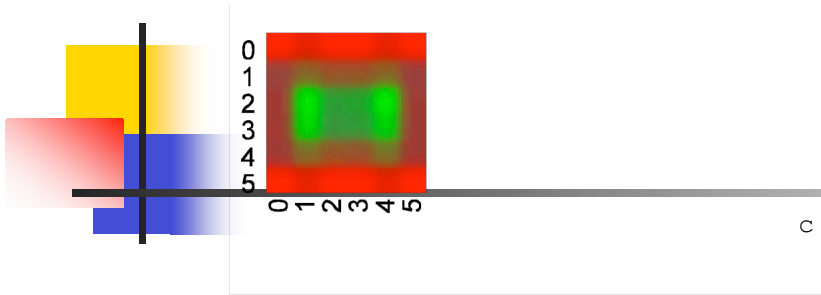
```

  if (ns_snd .ne. 0) then
    call snd_real(ns_snd, phijb, njb, ns_tag, info)
    c
    nmess = nmess + 1
    mess = mess + njb
  else
    if (j2.lt.0 .and. jbc.ne.0) then
      leak = 0.0
      do mi = 1, mmi
        m = mi + mio
        do lk = 1, nk
          k = k0 + sign(lk-1,k2)
          do i = 1, it
            phijbc(i,k,m,k3) = phijb(i,lk,mi)
            leak = leak + weta(m)*phijb(i,lk,mi)*di(i)*dk(k)
          end do
        end do
      end do
      leakage(3+j3) = leakage(3+j3) + leak
    else
      leak = 0.0
      do mi = 1, mmi
        m = mi + mio
        do lk = 1, nk
          k = k0 + sign(lk-1,k2)
          do i = 1, it
            leak = leak + weta(m)*phijb(i,lk,mi)*di(i)*dk(k)
          end do
        end do
      end do
      leakage(3+j3) = leakage(3+j3) + leak
    endif
  endif
endif

```



# Cluster 3: 1% of variance



Weight Clone ID

3.33143 sweep.f,sweep:449

3.04732 sweep.f,sweep:447

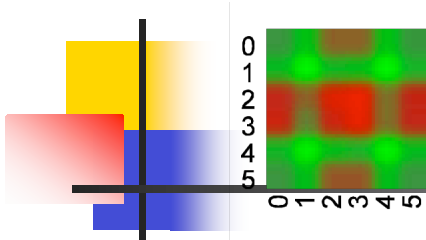
2.95163 sweep.f,sweep:445

```

c fixup i,j, & k if negative
  ifixed = 0
  continue
111
if (ti .lt. 0.0d+0) then
  dl = dl - ci
  ti = 1.0 / dl
  ql = ql - 0.5d+0*ci*phiir
  phi(i) = ql * ti
  ti = 0.0d+0
  if (tj .ne. 0.0d+0) tj=2.0d+0*phi(i) -phijb(i,lk,mi)
  if (tk .ne. 0.0d+0) tk=2.0d+0*phi(i) - phikb(i,j,mi)
  ifixed = 1
endif
if (tj .lt. 0.0d+0) then
  dl = dl - cj
  tj = 1.0 / dl
  ql = ql - 0.5d+0*cj*phijb(i,lk,mi)
  phi(i) = ql * tj
  tj = 0.0d+0
  if (tk .ne. 0.) tk = 2.0d+0*phi(i) - phikb(i,j,mi)
  if (ti .ne. 0.)ti = 2.0d+0*phi(i) - phiir
  ifixed = 1
  go to 111
endif
if (tk .lt. 0.0d+0) then
c   and it continues ... See cluster 4.

```

# Cluster 4: 0.8% of variance



Weight	Clone ID
4.48485	initialize.f,initxs:151
2.63445	sweep.f,sweep:456
2.51153	sweep.f,sweep:459
2.33486	initialize.f,initxs:148
1.89226	sweep.f,sweep:452

## ■ Initialize.f

```
do k = 1+k_l, kt-k_r
do j_g = max(1+j_l,jstart), min(jt_g-j_r,jstop)
do i_g = max(1+i_l,istart), min(it_g-i_r,istop)
  j = j_g - jstart + 1
  i = i_g - istart + 1
  Srcx(i,j,k) = 1.0d+0
end do
end do
end do
```

## ■ Sweep.f

```
if (tk .lt. 0.0d+0) then
  dl = dl - ck
  tk = 1.0 / dl
  ql = ql - 0.5d+0*ck*phikb(i,j,mi)
  phi(i) = ql * tk
  tk = 0.0d+0
  if (ti .ne. 0.0d+0) ti = 2.0d+0*phi(i) - phiir
  if (tj .ne. 0.0d+0) tj = 2.0d+0*phi(i)-phijb(i,lk,mi)
  ifixed = 1
  go to 111
endif
```



# Future Work

---

- Integrate with Eclipse PTP
- Explore other clustering algorithms
- Adapt to other performance data types
  - Communication data
  - Multiple metrics
- Parallelize clustering



# Conclusions

---

- As parallel systems and applications scale up, performance analysis tools must adapt their strategies to scale with the data.
- We can take advantage of statistical techniques like sampling and clustering to scalably manage and analyze large-scale performance data.



# Thanks!

---

## Questions?

This material is based on work supported by the National Science Foundation under Grant No. EIA-0216467 and Grant No. CNS-0421109, and the Department of Energy under Contract Nos. 03891-001-99-4G, 74837-001-03 49, 86192-001-04 49, and/or 12783-001-05 49 from the Los Alamos National Laboratory.

