# Soft Performance Analysis for Parallel and Distributed Programs

Hong-Linh Truong, Thomas Fahringer

Distributed and Parallel Systems Group

Institute of Computer Science

University of Innsbruck

{truong,tf}@dps.uibk.ac.at

http://dps.uibk.ac.at/projects/pma

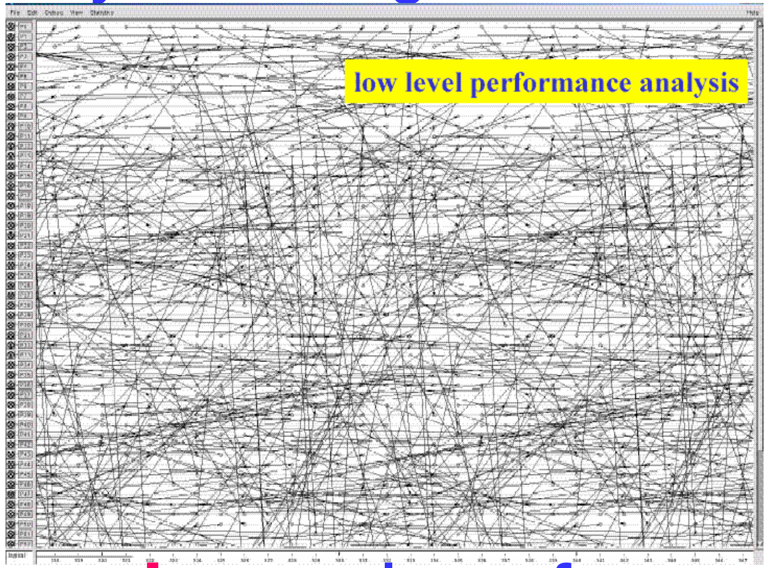
Euro-Par 05, Lisboa, 1st September, 2005

## Talk Outline

- Motivation
- Outline of soft performance analysis approach
- Performance score and similarity measure
- Some soft analysis techniques
- Conclusion and future work

## Motivation

- Lack of the specification and control of inexact parameters, commands and requests in existing performance analysis tools
- Performance tools do not interact with the user through high-level notation (e.g., words)
- Graphics techniques are very useful, but not suitable for performance analysis of large-scale and complex applications



Picture taken from a talk of D. Kranzlmueller (Uni. Linz)

- **Our approach:** apply soft computing, similarity measure, machine learning in performance analysis

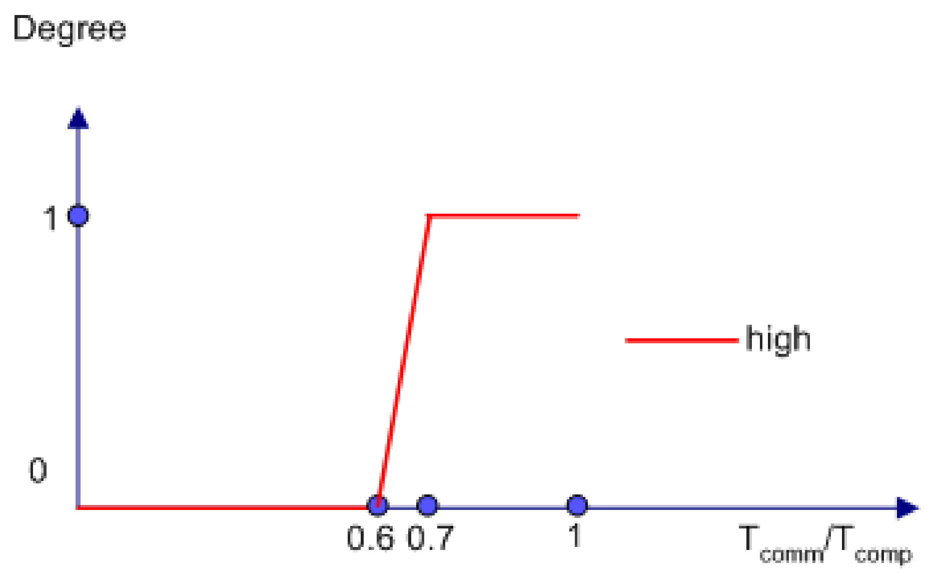
## Simple Example: Soft vs Hard Analysis

- Hard computing
  - Apply exact methods
  - Binary logic, crisp system, numerical analysis

  If Tcomm/Tcomp > 0.7 then r have high communication to computation ratio

- Soft Computing
  - Support imprecision and uncertainty
  - Computing with words

  If (Tcomm/Tcomp is high) then r have high communication to computation ratio with x degree



## Existing works

- ❖Fuzzy logic for performance monitoring, e.g. performance contracts (Pablo)
- ❖Using classification techniques based on machine learning, multivariate statistical techniques (e.g., done by Vetter and colleagues)
- ❖APART performance property characterizes specific negative performance behavior of code regions
- ❖Recent work applying data clustering in TAU (Uni. Oregon, to appear in SC05)

⇨ fuzzy logic has not been exploited in data analysis techniques, e.g., performance classification

⇨ not interact with the end user through high-level notation, e.g. linguistic query

# Outline of Soft Performance Analysis Approach

- Performance values are mapped into performance scores
- Performance characteristic terms are represented by a fuzzy set
  - A set of perf. characteristic terms describes possibilities of a metric
- To analyze the performance and interpret performance results with linguistic terms
- Similarity theory and machine learning: similarities and differences among performance data items
- Focuses of this talk
  - Conceptual framework: How can we apply soft computing into performance analysis
  - Interaction between performance tools and the user: Through high level notions and concepts expressed in linguistic expressions
  - Potential applications of soft performance analysis

## Preliminaries

- Performance data
  - A program consists of a set of (instrumented) code regions
  - Each code region is measured with a set of n metrics
- Performance experiment data used obtained from
  - 3DPIC, an MPI program, simulates the interaction of high intensity ultrashort laser pulses with plasma in three dimensional geometry
  - LAPW0 calculates the effective potential of the Kohn-Sham Eigen-Value problem, implemented in Fortran MPI
  - Stommel, OpenMP/MPI program, solves the 2d Stommel Model of Ocean Circulation using a Five-point stencil and Jacobi iteration.

# Performance Score

❖Performance score concept

- Map a value of metric m, v, into [0,1]. Performance score, s, of v is defined by

  s = ❍(v), ❍(v):[0,V] ✹ [0,1]

- ❍(v) is the membership function, V is the maximum value of m obtained from the base.
- Each code region is represented by a vector of scores
- Overall weighted average (OWA) for performance scores

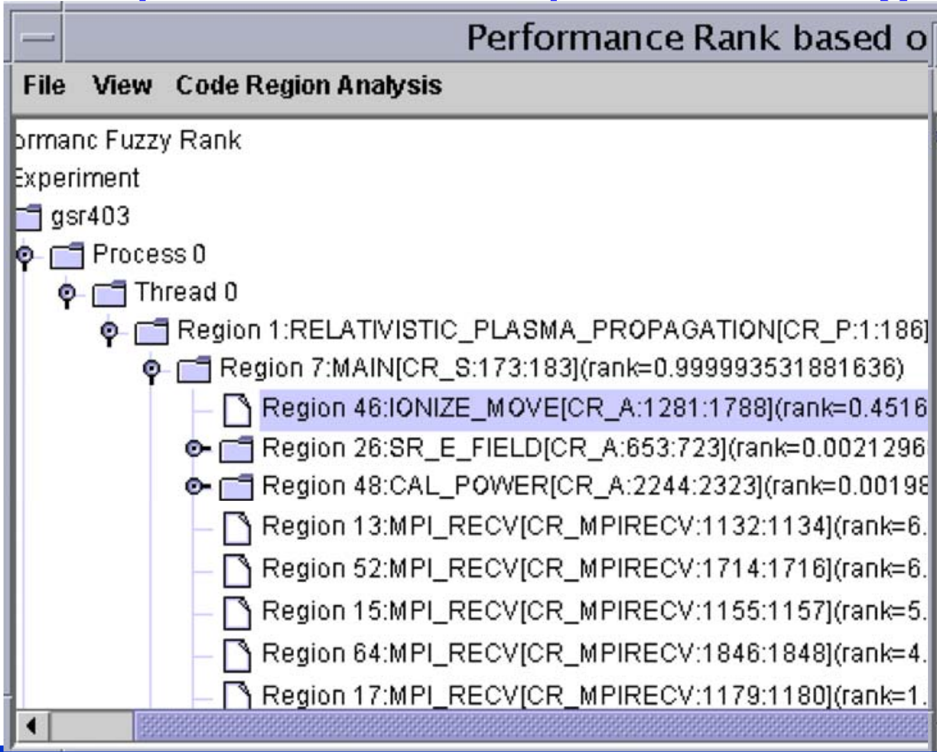
$$OWA(\vec{s}) = \frac{\sum_{i=1}^{n}(s_i * w_i)}{\sum_{i=1}^{n} w_i}$$

## Performance Score (cont.)

- The base is dependent on the scope of the analysis
  - Analysis can be done within a code region, a thread or the entire program
- [0,1]: 0 means lowest score, 1 means highest score
  - Semantics is defined by specific implementations
- Membership functions are also analysis-dependent
  - Examples: linear, S-function, etc.
- Performance score concept allows to normalize performance metrics but considering
  - The dynamics and flexibility
  - The uncertainty and imprecision
- Used  in dynamic tuning, ranking, clustering, etc.

# Ranking Analysis

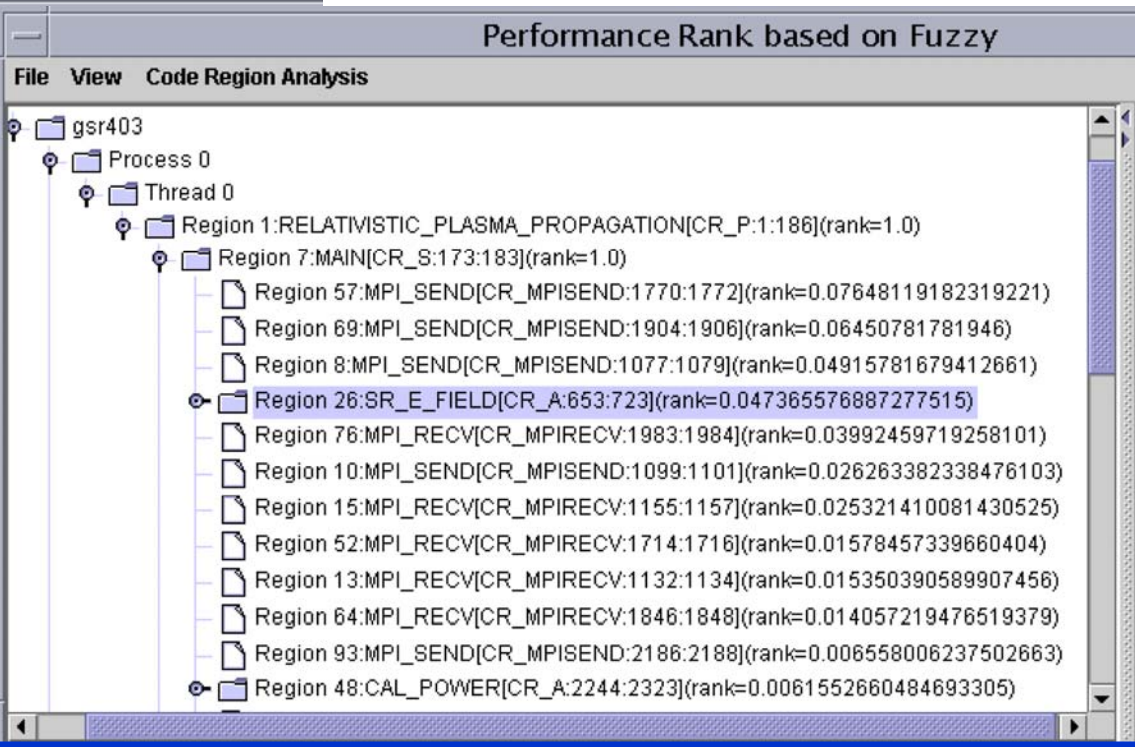
- Widely used in distinguishing significant and insignificant components
  - Which child code regions of a code region have strong impact on the performance of the parent?
- Ranking based on raw measurement value is difficult to interpret and compare the significance of the performance



Performance Rank based o

File View Code Region Analysis

```
ormanc Fuzzy Rank
Experiment
 gsr403
  Process 0
   Thread 0
    Region 1:RELATIVISTIC_PLASMA_PROPAGATION[CR_P:1:186]
     Region 7:MAIN[CR_S:173:183](rank=0.999993531881636)
      Region 46:IONIZE_MOVE[CR_A:1281:1788](rank=0.4516
      Region 26:SR_E_FIELD[CR_A:653:723](rank=0.0021296
      Region 48:CAL_POWER[CR_A:2244:2323](rank=0.00198
      Region 13:MPI_RECV[CR_MPIRECV:1132:1134](rank=6.
      Region 52:MPI_RECV[CR_MPIRECV:1714:1716](rank=6.
      Region 15:MPI_RECV[CR_MPIRECV:1155:1157](rank=5.
      Region 64:MPI_RECV[CR_MPIRECV:1846:1848](rank=4.
      Region 17:MPI_RECV[CR_MPIRECV:1179:1180](rank=1.
```



Performance Rank based on Fuzzy

File View Code Region Analysis

```
gsr403
 Process 0
  Thread 0
   Region 1:RELATIVISTIC_PLASMA_PROPAGATION[CR_P:1:186](rank=1.0)
    Region 7:MAIN[CR_S:173:183](rank=1.0)
     Region 57:MPI_SEND[CR_MPISEND:1770:1772](rank=0.07648119182319221)
     Region 69:MPI_SEND[CR_MPISEND:1904:1906](rank=0.06450781781946)
     Region 8:MPI_SEND[CR_MPISEND:1077:1079](rank=0.04915781679412661)
     Region 26:SR_E_FIELD[CR_A:653:723](rank=0.047365576887277515)
     Region 76:MPI_RECV[CR_MPIRECV:1983:1984](rank=0.03992459719258101)
     Region 10:MPI_SEND[CR_MPISEND:1099:1101](rank=0.026263382338476103)
     Region 15:MPI_RECV[CR_MPIRECV:1155:1157](rank=0.025321410081430525)
     Region 52:MPI_RECV[CR_MPIRECV:1714:1716](rank=0.01578457339660404)
     Region 13:MPI_RECV[CR_MPIRECV:1132:1134](rank=0.015350390589907456)
     Region 64:MPI_RECV[CR_MPIRECV:1846:1848](rank=0.014057219476519379)
     Region 93:MPI_SEND[CR_MPISEND:2186:2188](rank=0.006558006237502663)
     Region 48:CAL_POWER[CR_A:2244:2323](rank=0.0061552660484693305)
```

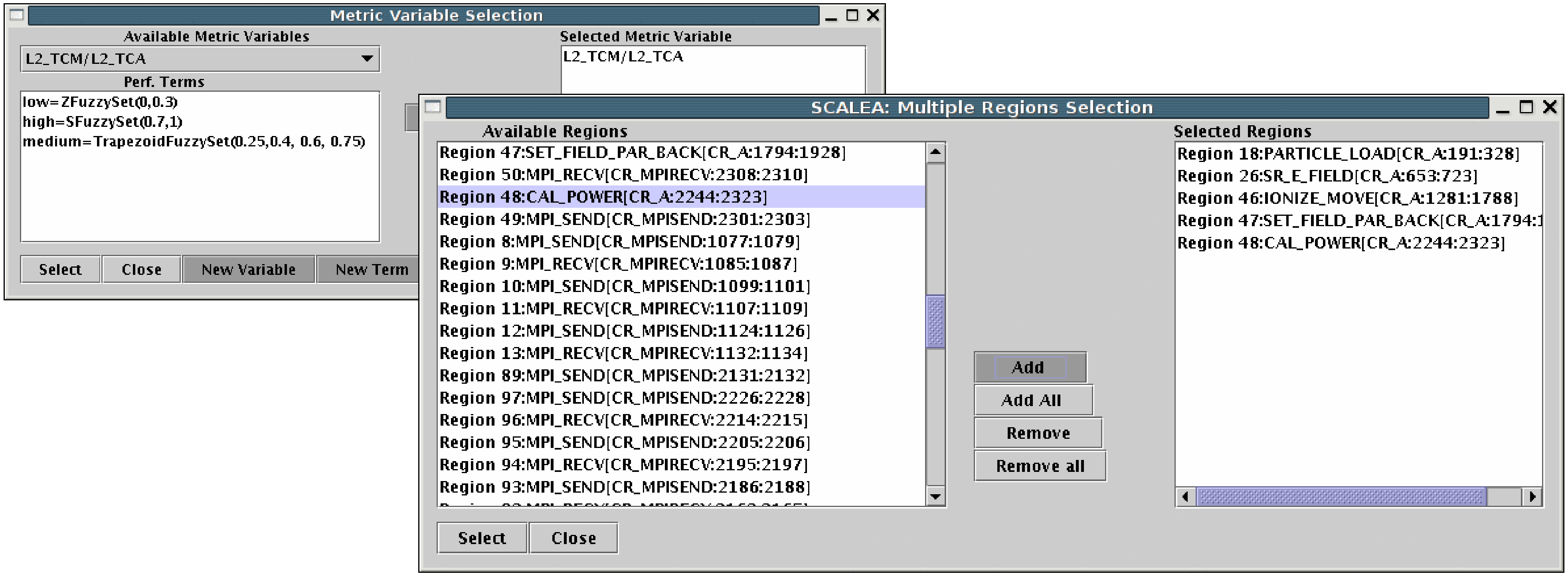
# Fuzzy-based Performance Classification

1. Define a set of *performance characteristic terms* T for a given metric

$$T = \{t_1, t_2, \ldots, t_n\}$$

2. A term is represented by a fuzzy set

3. Performance data are classified according to terms



Available Metric Variables: L2_TCM/L2_TCA

Selected Metric Variable: L2_TCM/L2_TCA

Perf. Terms:

```
low=ZFuzzySet(0,0.3)
high=SFuzzySet(0.7,1)
medium=TrapezoidFuzzySet(0.25,0.4, 0.6, 0.75)
```

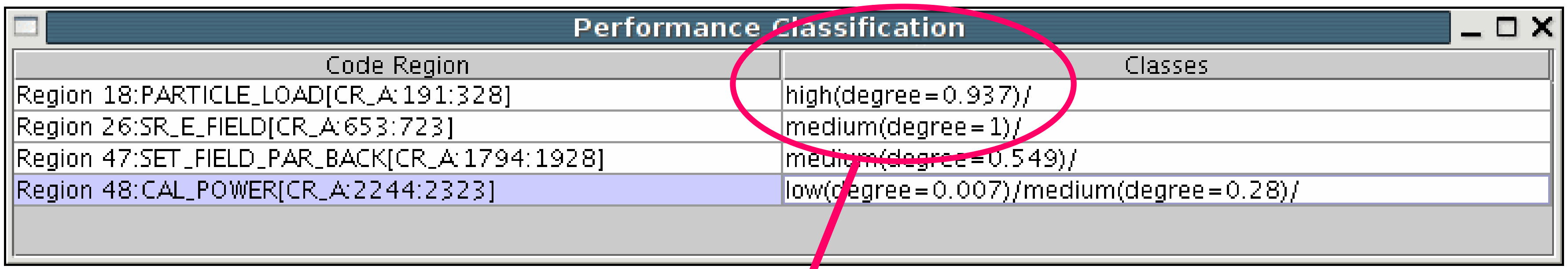
Available Regions:

```
Region 47:SET_FIELD_PAR_BACK[CR_A:1794:1928]
Region 50:MPI_RECV[CR_MPIRECV:2308:2310]
Region 48:CAL_POWER[CR_A:2244:2323]
Region 49:MPI_SEND[CR_MPISEND:2301:2303]
Region 8:MPI_SEND[CR_MPISEND:1077:1079]
Region 9:MPI_RECV[CR_MPIRECV:1085:1087]
Region 10:MPI_SEND[CR_MPISEND:1099:1101]
Region 11:MPI_RECV[CR_MPIRECV:1107:1109]
Region 12:MPI_SEND[CR_MPISEND:1124:1126]
Region 13:MPI_RECV[CR_MPIRECV:1132:1134]
Region 89:MPI_SEND[CR_MPISEND:2131:2132]
Region 97:MPI_SEND[CR_MPISEND:2226:2228]
Region 96:MPI_RECV[CR_MPIRECV:2214:2215]
Region 95:MPI_SEND[CR_MPISEND:2205:2206]
Region 94:MPI_RECV[CR_MPIRECV:2195:2197]
Region 93:MPI_SEND[CR_MPISEND:2186:2188]
```

Selected Regions:

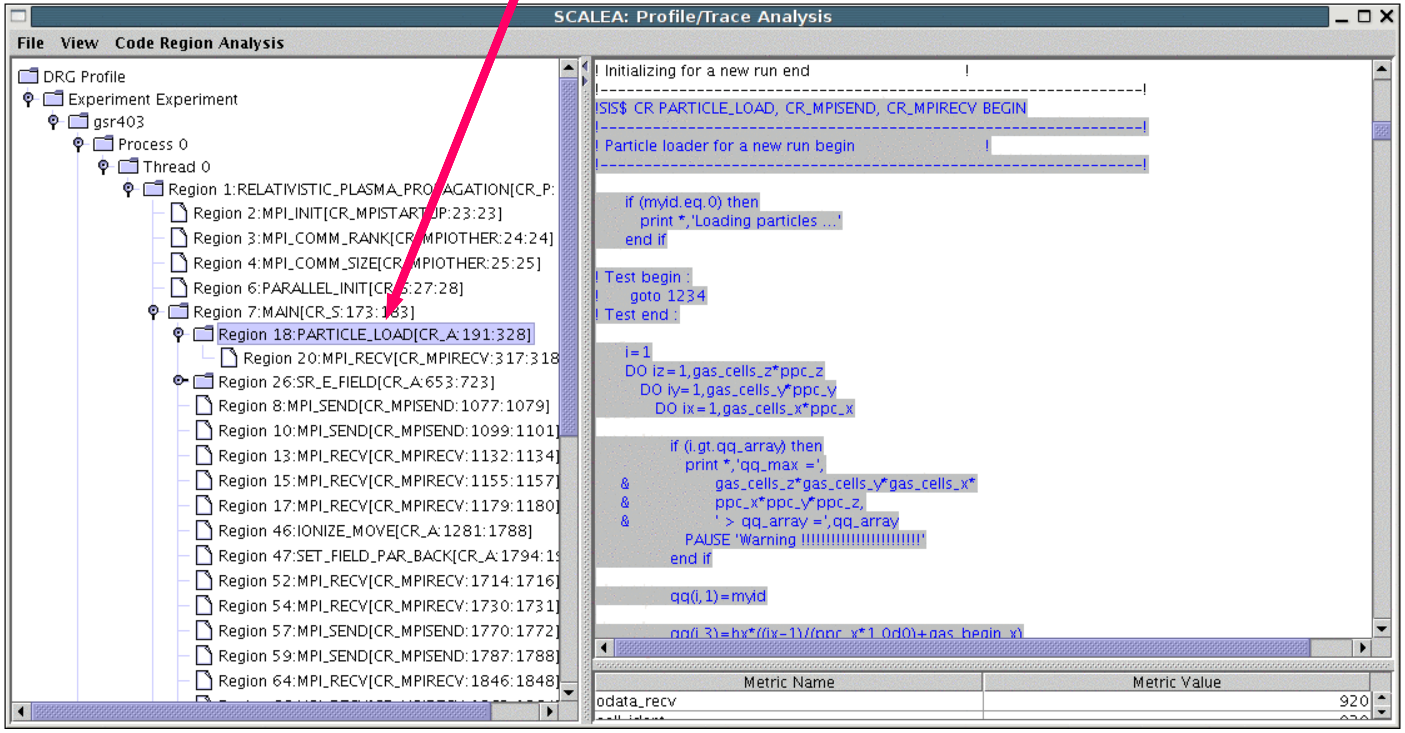
```
Region 18:PARTICLE_LOAD[CR_A:191:328]
Region 26:SR_E_FIELD[CR_A:653:723]
Region 46:IONIZE_MOVE[CR_A:1281:1788]
Region 47:SET_FIELD_PAR_BACK[CR_A:1794:1
Region 48:CAL_POWER[CR_A:2244:2323]
```

# Fuzzy-based Performance Classification (cont.)



Performance Classification

| Code Region | Classes |
|---|---|
| Region 18:PARTICLE_LOAD[CR_A:191:328] | high(degree=0.937)/ |
| Region 26:SR_E_FIELD[CR_A:653:723] | medium(degree=1)/ |
| Region 47:SET_FIELD_PAR_BACK[CR_A:1794:1928] | medium(degree=0.549)/ |
| Region 48:CAL_POWER[CR_A:2244:2323] | low(degree=0.007)/medium(degree=0.28)/ |



SCALEA: Profile/Trace Analysis

File View Code Region Analysis

```
DRG Profile
  Experiment Experiment
    gsr403
      Process 0
        Thread 0
          Region 1:RELATIVISTIC_PLASMA_PROPAGATION[CR_P:
            Region 2:MPI_INIT[CR_MPISTARTUP:23:23]
            Region 3:MPI_COMM_RANK[CR_MPIOTHER:24:24]
            Region 4:MPI_COMM_SIZE[CR_MPIOTHER:25:25]
            Region 6:PARALLEL_INIT[CR_S:27:28]
            Region 7:MAIN[CR_S:173:183]
              Region 18:PARTICLE_LOAD[CR_A:191:328]
                Region 20:MPI_RECV[CR_MPIRECV:317:318
              Region 26:SR_E_FIELD[CR_A:653:723]
              Region 8:MPI_SEND[CR_MPISEND:1077:1079]
              Region 10:MPI_SEND[CR_MPISEND:1099:1101]
              Region 13:MPI_RECV[CR_MPIRECV:1132:1134]
              Region 15:MPI_RECV[CR_MPIRECV:1155:1157]
              Region 17:MPI_RECV[CR_MPIRECV:1179:1180]
              Region 46:IONIZE_MOVE[CR_A:1281:1788]
              Region 47:SET_FIELD_PAR_BACK[CR_A:1794:1
              Region 52:MPI_RECV[CR_MPIRECV:1714:1716]
              Region 54:MPI_RECV[CR_MPIRECV:1730:1731]
              Region 57:MPI_SEND[CR_MPISEND:1770:1772]
              Region 59:MPI_SEND[CR_MPISEND:1787:1788]
              Region 64:MPI_RECV[CR_MPIRECV:1846:1848]
```

```
! Initializing for a new run end                    !
!-----------------------------------------------------------------!
!SIS$ CR PARTICLE_LOAD, CR_MPISEND, CR_MPIRECV BEGIN
!-----------------------------------------------------------------!
! Particle loader for a new run begin                       !
!-----------------------------------------------------------------!

      if (myid.eq.0) then
         print *,'Loading particles ...'
      end if

! Test begin :
!      goto 1234
! Test end :

      i=1
      DO iz=1,gas_cells_z*ppc_z
         DO iy=1,gas_cells_y*ppc_y
            DO ix=1,gas_cells_x*ppc_x

               if (i.gt.qq_array) then
                  print *,'qq_max =',
     &                 gas_cells_z*gas_cells_y*gas_cells_x*
     &                 ppc_x*ppc_y*ppc_z,
     &                 ' > qq_array =',qq_array
                  PAUSE 'Warning !!!!!!!!!!!!!!!!!!!!!!!!'
               end if

               qq(i,1)=myid

               qq(i,3)=hx*((ix-1)/(ppc_x*1.0d0)+gas_begin_x)
```

| Metric Name | Metric Value |
|---|---|
| odata_recv | 920 |

## Fuzzy-based Performance Search

- Existing performance tools
  - Do not offer the possibility of search performance data with linguistic query
- PERFormance Query Language based on fuzzy logic (PERFQL)
  - Performance search based on linguistic expressions
  - Easily to define/understand queries

```
<PERFQL_Statement> ::= <PERFQL_Expr> | <PERFQL_Statement>
                                    OR  <PERFQL_Expr>
<PERFQL_Expr >          ::=<PERFQL_Term> | <PERFQL_Expr>
                                    AND <PERFQL_Term>
<PERFQL_Term>      ::= (<METRIC_Expr> is <F_Expr>)
```

**Metric or Metric Expression**

**wtime**

**L2_TCM/L2_TCA**

**odata_send/wtime**

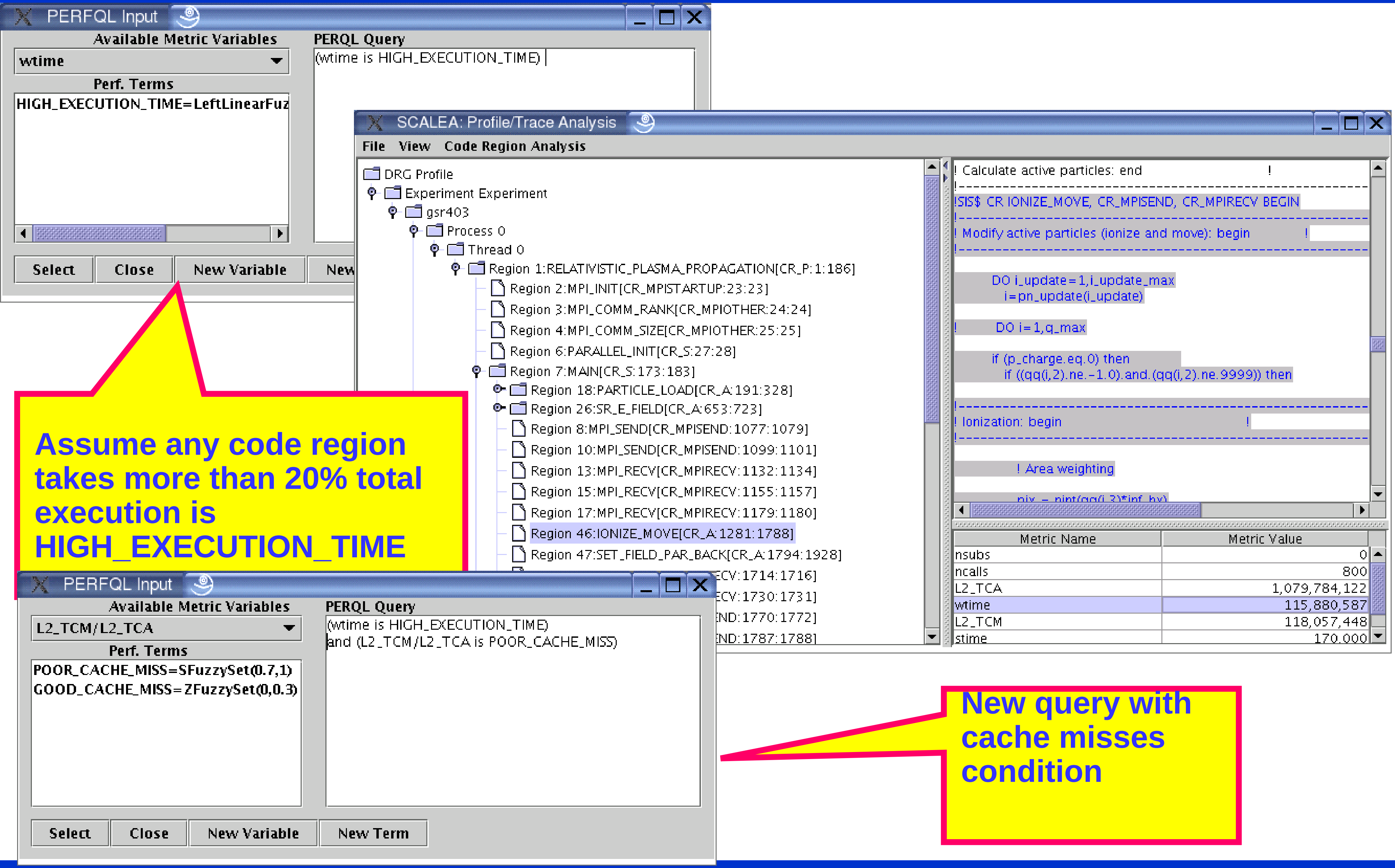
**Fuzzy Expression**

**HIGH_EXECUTION_TIME**

**very HIGH_EXECUTION_TIME**

**slightly POOR_SEND_OVERHEAD**

# Fuzzy-based Performance  Search (cont.)



PERFQL Input

Available Metric Variables: wtime

Perf. Terms

HIGH_EXECUTION_TIME=LeftLinearFuz

PERQL Query

(wtime is HIGH_EXECUTION_TIME) |

Select | Close | New Variable | New

SCALEA: Profile/Trace Analysis

File  View  Code Region Analysis

```
DRG Profile
  Experiment Experiment
    gsr403
      Process 0
        Thread 0
          Region 1:RELATIVISTIC_PLASMA_PROPAGATION[CR_P:1:186]
            Region 2:MPI_INIT[CR_MPISTARTUP:23:23]
            Region 3:MPI_COMM_RANK[CR_MPIOTHER:24:24]
            Region 4:MPI_COMM_SIZE[CR_MPIOTHER:25:25]
            Region 6:PARALLEL_INIT[CR_S:27:28]
            Region 7:MAIN[CR_S:173:183]
              Region 18:PARTICLE_LOAD[CR_A:191:328]
              Region 26:SR_E_FIELD[CR_A:653:723]
              Region 8:MPI_SEND[CR_MPISEND:1077:1079]
              Region 10:MPI_SEND[CR_MPISEND:1099:1101]
              Region 13:MPI_RECV[CR_MPIRECV:1132:1134]
              Region 15:MPI_RECV[CR_MPIRECV:1155:1157]
              Region 17:MPI_RECV[CR_MPIRECV:1179:1180]
              Region 46:IONIZE_MOVE[CR_A:1281:1788]
              Region 47:SET_FIELD_PAR_BACK[CR_A:1794:1928]
              ECV:1714:1716]
              ECV:1730:1731]
              ND:1770:1772]
              ND:1787:1788]
```

```
! Calculate active particles: end                    !
!----------------------------------------------------------
!SIS$ CR IONIZE_MOVE, CR_MPISEND, CR_MPIRECV BEGIN
!----------------------------------------------------------
! Modify active particles (ionize and move): begin      !
!----------------------------------------------------------

      DO i_update=1,i_update_max
        i=pn_update(i_update)

!      DO i=1,q_max

      if (p_charge.eq.0) then
        if ((qq(i,2).ne.-1.0).and.(qq(i,2).ne.9999)) then

!----------------------------------------------------------
! Ionization: begin                                  !
!----------------------------------------------------------

          ! Area weighting

          nix = nint(qq(i,3)*inf_hx)
```

| Metric Name | Metric Value |
|---|---|
| nsubs | 0 |
| ncalls | 800 |
| L2_TCA | 1,079,784,122 |
| wtime | 115,880,587 |
| L2_TCM | 118,057,448 |
| stime | 170,000 |

PERFQL Input

Available Metric Variables: L2_TCM/L2_TCA

Perf. Terms

POOR_CACHE_MISS=SFuzzySet(0.7,1)
GOOD_CACHE_MISS=ZFuzzySet(0,0.3)

PERQL Query

(wtime is HIGH_EXECUTION_TIME)
and (L2_TCM/L2_TCA is POOR_CACHE_MISS)

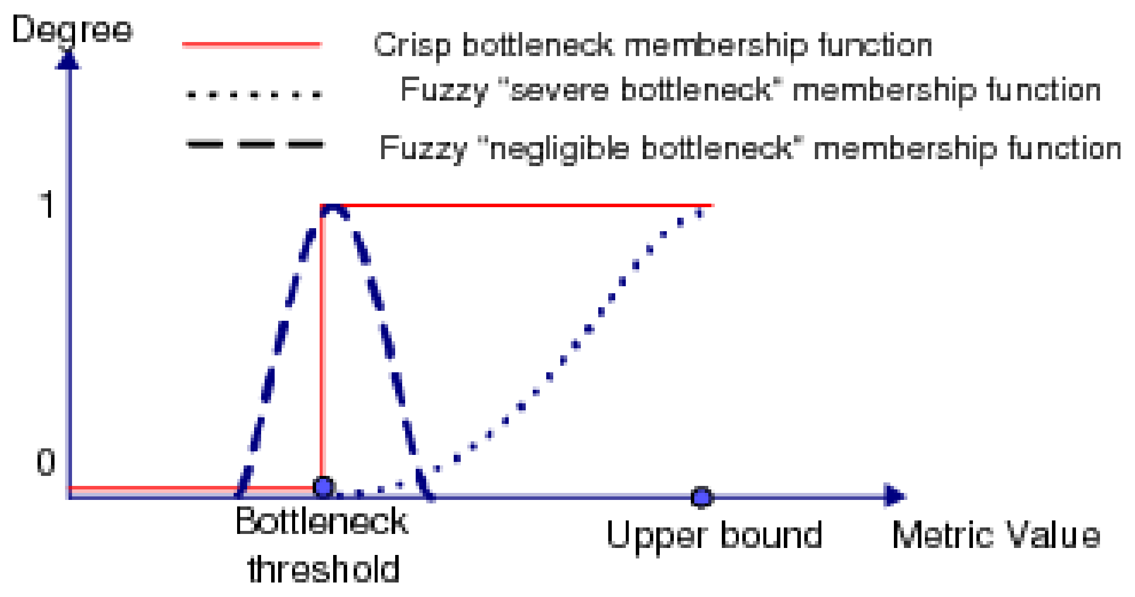
Select | Close | New Variable | New Term

**Assume any code region takes more than 20% total execution is HIGH_EXECUTION_TIME**

**New query with cache misses condition**

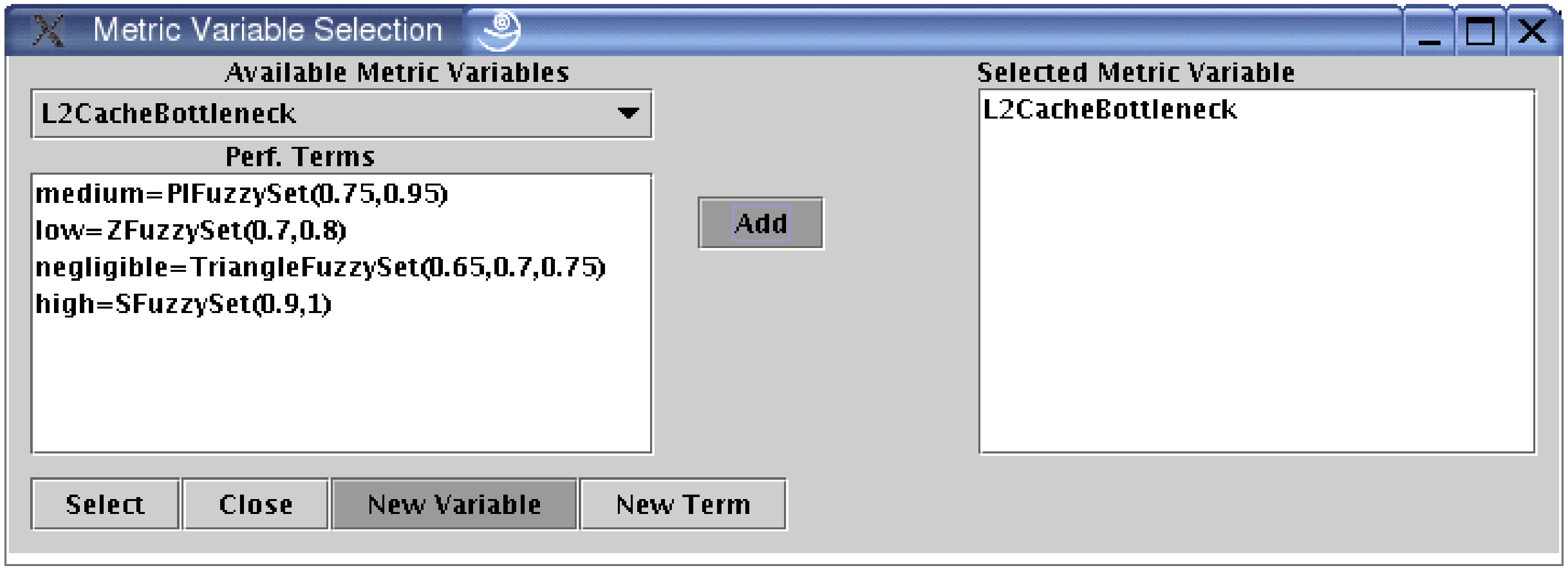
# Fuzzy Approach to Bottleneck Search

1. Using fuzzy sets to represent *bottleneck conditions*
2. Using fuzzy sets to represent *negligible bottlenecks*



- Search results
  - Indicate the *degree of bottleneck*
    - We can use the degree of bottleneck for further tasks
  - Locate *negligible bottlenecks*
    - We may not find any bottlenecks because the condition is not exact

# Bottleneck Search: Simple Example

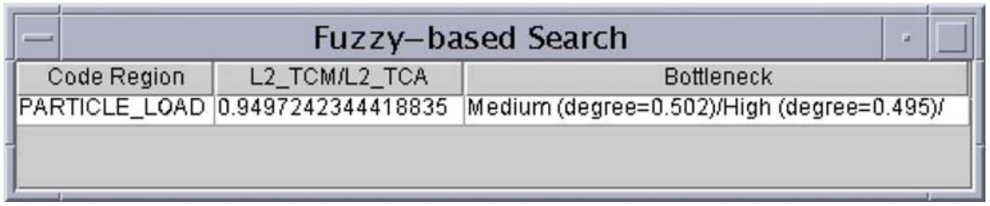


L2CacheBottleneck

medium=PIFuzzySet(0.75,0.95)
low=ZFuzzySet(0.7,0.8)
negligible=TriangleFuzzySet(0.65,0.7,0.75)
high=SFuzzySet(0.9,1)

Selected: L2CacheBottleneck

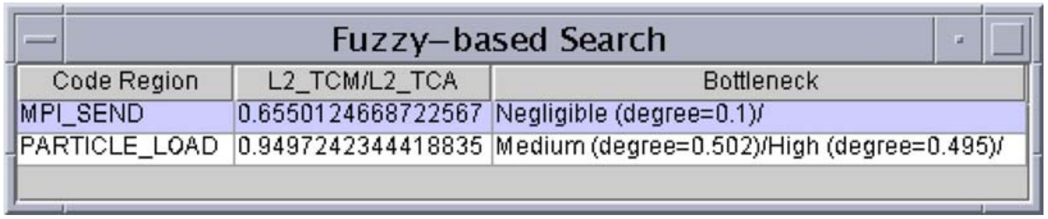
- Search for low, medium and high *degree of bottleneck*



Fuzzy–based Search

| Code Region | L2_TCM/L2_TCA | Bottleneck |
|---|---|---|
| PARTICLE_LOAD | 0.9497242344418835 | Medium (degree=0.502)/High (degree=0.495)/ |

- Search also *negligible bottlenecks*



Fuzzy–based Search

| Code Region | L2_TCM/L2_TCA | Bottleneck |
|---|---|---|
| MPI_SEND | 0.6550124668722567 | Negligible (degree=0.1)/ |
| PARTICLE_LOAD | 0.9497242344418835 | Medium (degree=0.502)/High (degree=0.495)/ |

## Performance Similarity Measure

- Problems:
  - Difficult to observe and perceive the performance similarity and difference through complex visualization
- Performance similarity measure indicates the performance similarity among code regions and among experiment factors

$$sim(o_i,o_j) \rightarrow [0,1]$$

  - 0 denotes complete dissimilarity and 1 denotes complete similarity

# Performance Similarity Measure

- Performance similarity measure for code regions
  1. Using performance score concept to determine performance scores of region summaries $rs_i$ and $rs_j$. Each rs is represented as a vector of n performance scores
  2. Determining distance measure between $rs_i$ and $rs_j$. For example,

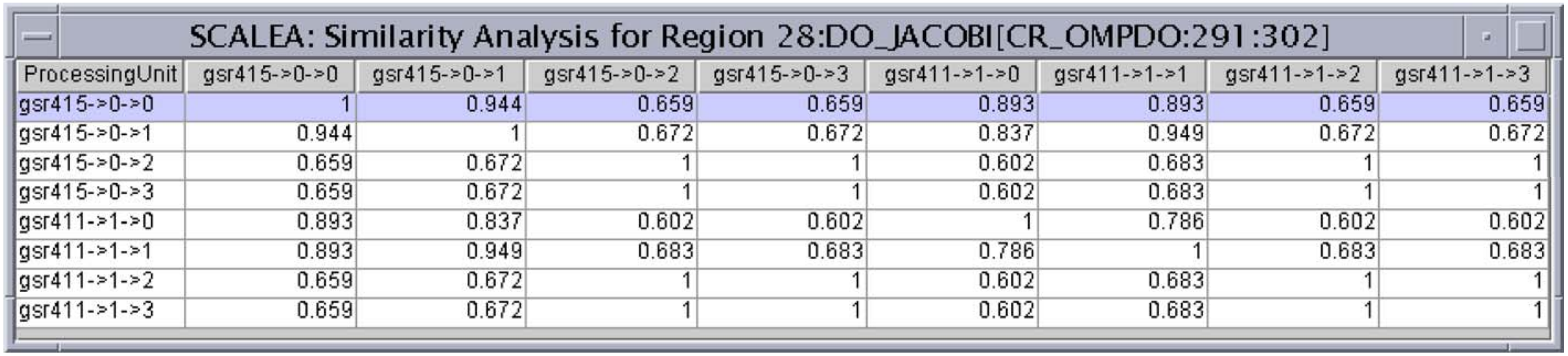
$$d_{ij} = \sqrt{\sum_{l=1}^{n} (s_{il} - s_{jl})^2}$$

  3. Determining performance similarity between two code regions

$$sim_{ij}(rs_i, rs_j) = 1 - d_{ij}$$

# Performance Similarity Analysis (cont.)

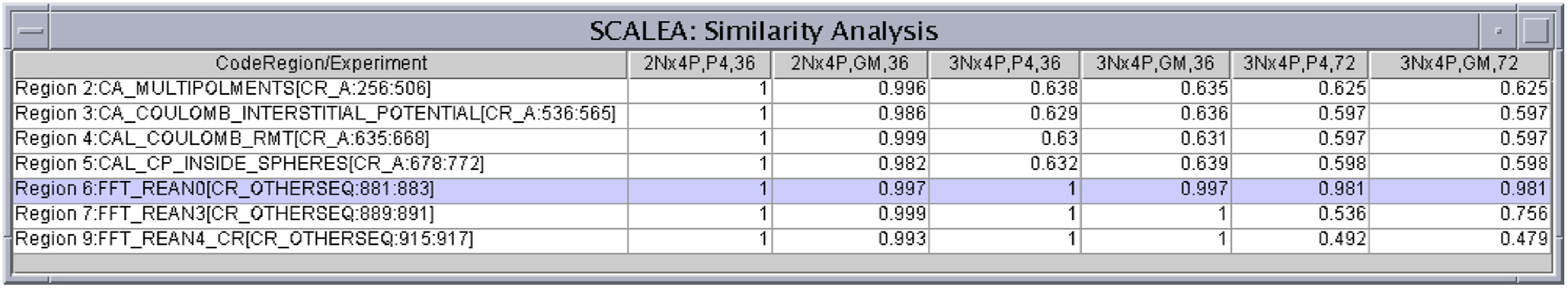
- Stommel:
  - Similarity measure for cache accesses of Stommel application



SCALEA: Similarity Analysis for Region 28:DO_JACOBI[CR_OMPDO:291:302]

| ProcessingUnit | gsr415->0->0 | gsr415->0->1 | gsr415->0->2 | gsr415->0->3 | gsr411->1->0 | gsr411->1->1 | gsr411->1->2 | gsr411->1->3 |
|---|---|---|---|---|---|---|---|---|
| gsr415->0->0 | 1 | 0.944 | 0.659 | 0.659 | 0.893 | 0.893 | 0.659 | 0.659 |
| gsr415->0->1 | 0.944 | 1 | 0.672 | 0.672 | 0.837 | 0.949 | 0.672 | 0.672 |
| gsr415->0->2 | 0.659 | 0.672 | 1 | 1 | 0.602 | 0.683 | 1 | 1 |
| gsr415->0->3 | 0.659 | 0.672 | 1 | 1 | 0.602 | 0.683 | 1 | 1 |
| gsr411->1->0 | 0.893 | 0.837 | 0.602 | 0.602 | 1 | 0.786 | 0.602 | 0.602 |
| gsr411->1->1 | 0.893 | 0.949 | 0.683 | 0.683 | 0.786 | 1 | 0.683 | 0.683 |
| gsr411->1->2 | 0.659 | 0.672 | 1 | 1 | 0.602 | 0.683 | 1 | 1 |
| gsr411->1->3 | 0.659 | 0.672 | 1 | 1 | 0.602 | 0.683 | 1 | 1 |

- LAPW0:
  - Similarity measure based on wallclock time



SCALEA: Similarity Analysis

| CodeRegion/Experiment | 2Nx4P,P4,36 | 2Nx4P,GM,36 | 3Nx4P,P4,36 | 3Nx4P,GM,36 | 3Nx4P,P4,72 | 3Nx4P,GM,72 |
|---|---|---|---|---|---|---|
| Region 2:CA_MULTIPOLMENTS[CR_A:256:506] | 1 | 0.996 | 0.638 | 0.635 | 0.625 | 0.625 |
| Region 3:CA_COULOMB_INTERSTITIAL_POTENTIAL[CR_A:536:565] | 1 | 0.986 | 0.629 | 0.636 | 0.597 | 0.597 |
| Region 4:CAL_COULOMB_RMT[CR_A:635:668] | 1 | 0.999 | 0.63 | 0.631 | 0.597 | 0.597 |
| Region 5:CAL_CP_INSIDE_SPHERES[CR_A:678:772] | 1 | 0.982 | 0.632 | 0.639 | 0.598 | 0.598 |
| Region 6:FFT_REAN0[CR_OTHERSEQ:881:883] | 1 | 0.997 | 1 | 0.997 | 0.981 | 0.981 |
| Region 7:FFT_REAN3[CR_OTHERSEQ:889:891] | 1 | 0.999 | 1 | 1 | 0.536 | 0.756 |
| Region 9:FFT_REAN4_CR[CR_OTHERSEQ:915:917] | 1 | 0.993 | 1 | 1 | 0.492 | 0.479 |

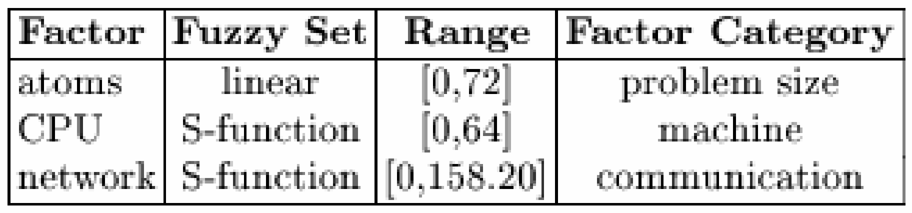
# Performance Similarity Analysis (cont.)

- Performance similarity measure for experiment factors

  Given a set of controllable factors $F=\{f_1,f_2, \ldots,f_n\}$ and given experiments $e_i$ and $e_j$

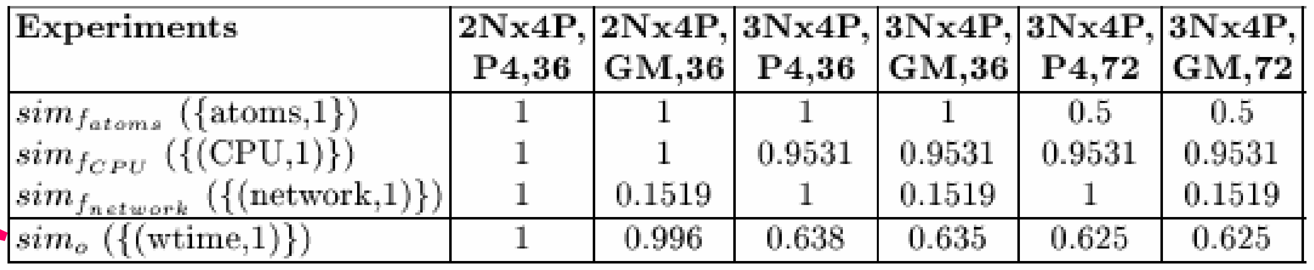
  1. Factor f is described by a membership function
  2. Determine similarity measure between f of $e_i$ and $e_j$, $sim_f(e_i,e_j)$
  3. Analyze relations among similarity measures for code regions and experiment factors

- LAPW0



| Factor | Fuzzy Set | Range | Factor Category |
|---|---|---|---|
| atoms | linear | [0,72] | problem size |
| CPU | S-function | [0,64] | machine |
| network | S-function | [0,158.20] | communication |

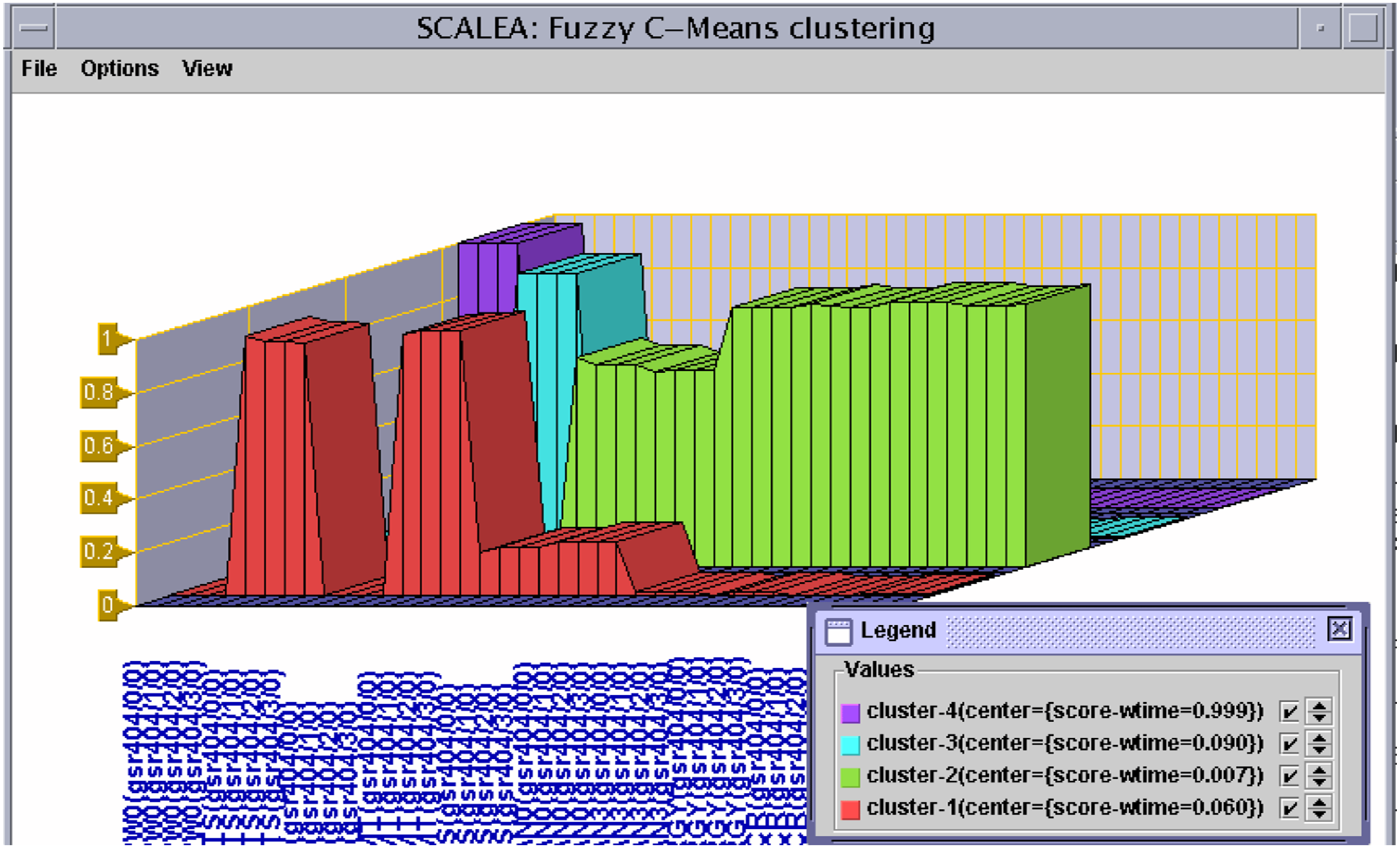
**Similarity analysis for CA_MUTIPOLMENTS region**



| Experiments | 2Nx4P, P4,36 | 2Nx4P, GM,36 | 3Nx4P, P4,36 | 3Nx4P, GM,36 | 3Nx4P, P4,72 | 3Nx4P, GM,72 |
|---|---|---|---|---|---|---|
| $sim_{f_{atoms}}$ ({atoms,1}) | 1 | 1 | 1 | 1 | 0.5 | 0.5 |
| $sim_{f_{CPU}}$ ({(CPU,1)}) | 1 | 1 | 0.9531 | 0.9531 | 0.9531 | 0.9531 |
| $sim_{f_{network}}$ ({(network,1)}) | 1 | 0.1519 | 1 | 0.1519 | 1 | 0.1519 |
| $sim_o$ ({(wtime,1)}) | 1 | 0.996 | 0.638 | 0.635 | 0.625 | 0.625 |

# Fuzzy C-Means Clustering

- 3D PIC executed with 4 processes



| | |
|---|---|
| 🟪 | cluster-4(center={score-wtime=0.999}) |
| 🟦 | cluster-3(center={score-wtime=0.090}) |
| 🟩 | cluster-2(center={score-wtime=0.007}) |
| 🟥 | cluster-1(center={score-wtime=0.060}) |

## Other potential applications of soft performance analysis techniques

- Decision making in dynamic performance tuning
  - Dynamic performance tuning tools: MATE (UAB), Active Harmony (J. Hollingsworth)
  - Automatically replacing components, selecting different implementations based on  performance scores and performance similarity measures
- Performance data collection/reduction
  - Rules based on crisp-condition can be replaced by fuzzy rules based on performance scores
- etc.

## Conclusion and Future Work

- Contributions: we proposed the soft performance analysis approach
  - Provide flexible, scalable techniques for analyzing and comparing the performance of parallel and distributed applications
  - Interact with the user through high-level notation
  - Aim to support automatic performance analysis
- However, soft performance analysis is just at an early stage
  - Not everything discussed has been fully implemented
- What should be done next
  - So far, we have just focused on conceptual framework, not on how to select membership and distance functions
    - Study the selection of membership and distance functions
  - Apply soft performance analysis for dynamic performance tuning, autonomic computing
  - Linguistic variables and fuzzy rules for SLAs (service level agreements) in the Grid