Outline

- Motivating Example
- Basic Issues in SPE
- A Methodology for SPE in C/S
- Clisspe: A Language for SPE in C/S
- Benchmarking and Data Collection for SPE
- A Case Study
- Concluding Remarks
Motivating Example

• Recruitment and Training System (RTS)
  – applicants go to recruitment centers spread all over the country.
  – a guiding counselor interviews the applicant and tries to match the applicant skills with the agency’s desired skills.
  – accepted applicants are recruited and are assigned to one or more training classes where they will acquire the skills needed for the job.
Motivating Example

• Current system:
  – centralized,
  – database and application on a mainframe,
  – line-oriented user interface,
  – expensive to maintain and upgrade (some programs are 20 years old),
  – does not scale well with the number of users.
Motivating Example

- The system will be migrated to a C/S environment with a GUI running at the clients.
- Application modules will be stored at different application servers and will be executed as needed.
- Several DB servers will store portions of the DB.
Motivating Example DB

Applicant <has> Skill <requires> Course

Applicant <needs> Section

Skill <develops> Course <taught>

Motivating Example DB Tables

**Applicant**

<table>
<thead>
<tr>
<th>SSN</th>
<th>Name</th>
<th>StreetAddress</th>
<th>City</th>
<th>Zip</th>
<th>Phone</th>
<th>Education</th>
<th>...</th>
</tr>
</thead>
</table>

**Skill**

<table>
<thead>
<tr>
<th>SkillCode</th>
<th>SkillName</th>
<th>SkillDescription</th>
<th>SkillMinVal</th>
<th>SkillMaxVal</th>
</tr>
</thead>
</table>

**Course**

<table>
<thead>
<tr>
<th>CourseNum</th>
<th>CourseName</th>
<th>NumHours</th>
<th>Description</th>
</tr>
</thead>
</table>

**Section**

<table>
<thead>
<tr>
<th>CourseNum</th>
<th>SectionNum</th>
<th>StartDate</th>
<th>DayTime</th>
<th>Location</th>
<th>MaxCap</th>
</tr>
</thead>
</table>
Motivating Example DB Tables

<table>
<thead>
<tr>
<th>Enrollment</th>
<th>CourseDevelopsSkill</th>
</tr>
</thead>
<tbody>
<tr>
<td>CourseNum</td>
<td>SectionNum</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ApplicantHasSkill</th>
<th>CourseRequiresSkill</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSN</td>
<td>SkillCode</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ApplicantNeedsSkill</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>SSN</td>
<td>SkillCode</td>
</tr>
</tbody>
</table>

Motivating Example

SPE Questions

- How “thin” should the client software be?
  - How much work should be done at the client versus at the application server?
- How should the DB be distributed?
  - how many DB servers we need?
  - which tables should be stored in each DB server?
  - should tables be partitioned by rows and stored at different DB servers?
  - should tables be replicated and how?
Motivating Example
SPE Questions

• What kind of hardware and OS platform should be used for the application servers?
• What kind of hardware and OS platform should be used for the DB servers?
• How many DB and application servers are needed and where should they be located?
• What type of networking technology and connectivity should be used?

Motivating Example
SPE Questions

• What DBMS should be used to support the DB server?
• What indexes should be created on the various DB tables?
Motivating Example
SPE General Questions

• Will the new application meet the service level requirements?

• How many clients will be supported and at what cost?

Outline

• Motivating Example
• Basic Issues in SPE
• A Methodology for SPE in C/S
• Clisspe: A Language for SPE in C/S
• Benchmarking and Data Collection for SPE
• A Case Study
• Concluding Remarks
Traditional Software Development Life Cycle

- Common approach:
  - consider Functional Requirements only during development and check Performance Requirements at the end.
  - fix the system if performance is not good!
- Problem:
  - it is very costly and time consuming to fix the problem after the system is ready!
  - fixing the problem may imply in major software rewrites.
Integrating Software Performance Engineering Into the Software Development Life Cycle

The Software Development Life Cycle and Inputs to SPE

- service levels (response times, throughputs, etc)
- hardware/software base (client and software platforms, networking technologies, DBMSs)

- mapping of software modules to C/S architecture
- database design
- networking topology

- number of I/Os and DBMS calls per transaction
- estimates of CPU demand
- estimates of network traffic

- refined I/O and CPU demand estimates
- refined network traffic demand estimates

- measured I/O, CPU, and network service demand estimates
Main Steps in SPE for C/S Systems

• Understand the Environment:
  – determine the critical transactions using the 80/20 rule: 20% of transactions that are likely to use 80% of the resources.
  – determine the cost and technology constraints
    (e.g., what client and server H/S platforms should be used, what networking technologies are to be used)
  – determine the service levels for the critical transactions.
  – determine the base C/S architecture.
  – is there a mainframe version of the application?
Main Steps in SPE for C/S Systems

• Characterize the Workload:
  – for each critical transaction, find:
    • estimated workload intensity (if there is a mainframe based system, get these from there).
    • estimated service demands for:
      – client and server processors
      – client and server disks
      – LAN segments
      – WANs
      – routers

Main Steps in SPE for C/S Systems

• Build a Performance Model:
  – build a performance (typically a queuing network model) that corresponds to the complete system.

• Solve the Performance Model:
  – obtain response times and throughputs per transaction
  – obtain a break down of response time per device
  – determine bottlenecks.
Main Steps in SPE for C/S Systems

• Performance Assessment:
  – compare estimated performance metrics with service levels.
  – if performance is poor, verify where transactions spend most of their time and give feedback to system designers to cause changes in:
    • software architecture
    • work distribution between clients and servers
    • database allocation to servers
    • allocation of servers to networks
    • other

Outline

• Motivating Example
• Basic Issues in SPE
• A Methodology for SPE in C/S
• Clisspe: A Language for SPE in C/S
• Benchmarking and Data Collection for SPE
• A Case Study
• Concluding Remarks
Clisspe: a Client Server Software Performance Engineering Language

- Declaration Section
- Mapping Section
- Transaction Section
Clisspe Declaration Section

- clients and client types
- servers and server types
- disks and disk types
- database management systems
- database tables
- networks and network types
- transactions
- remote procedure calls
- numeric constants

Example of Clisspe Declaration Section for RTS System

```clisspe
model rts

declaration ! declaration section for RTS example
dbms Oracle page_size= 2048;

! client types and client declarations
client_type Pentium120 specint92= 133 specfp92= 99
    IO_benchmark (a= 0.001, b= 0.5);
client gc_DC type= Pentium120 number= 100
    disk dsk01 seek= 0.01 latency= 0.00833 xfer_rate= 10;
client gc_LA type= Pentium120 number= 50
    disk dsk01 seek= 0.01 latency= 0.00833 xfer_rate= 10;
client gc_WA type= Pentium120 number= 80
    disk dsk01 seek= 0.01 latency= 0.00833 xfer_rate= 10;
```

© 1997 Daniel A. Menascé
Example of Clisspe Declaration Section for RTS System (continued)

! server types
server_type IBM_RS/6000_M43P133 ! DB server type
  specint_92= 176.4 specfp_92= 156.5
  IO_benchmark (a= 0.00005, b= 0.06) ;

server_type IBM_RS/6000_M43P120 ! application server type
  specint_92= 157.9 specfp_92= 139.2
  IO_benchmark (a= 0.00005, b= 0.06) ;

Example of Clisspe Declaration Section for RTS System (continued)

! application servers
server appl_DC type= IBM_RS/6000_M43P120
  num_CPUs= 1
  disk dsk01 seek= 0.015 latency= 0.00833 xfer_rate= 10;

server appl_LA type= IBM_RS/6000_M43P120
  num_CPUs= 1
  disk dsk01 seek= 0.015 latency= 0.00833 xfer_rate= 10;

server appl_WA type= IBM_RS/6000_M43P120
  num_CPUs= 1
  disk dsk01 seek= 0.015 latency= 0.00833 xfer_rate= 10;
Example of Clisspe Declaration Section for RTS System (continued)

! database servers
disk_type ServerDisk seek= 0.015 latency= 0.00833 xfer_rate= 10;

server DBserver_DC type= IBM_RS/6000_M43P133
dbms= Oracle DB_BuffSize= 8192 num_CPUs= 2
disk dsk01 type= ServerDisk disk dsk02 type= ServerDisk
disk dsk03 type= ServerDisk;

server DBserver_LA type= IBM_RS/6000_M43P133
dbms= Oracle DB_BuffSize= 8192 num_CPUs= 2
disk dsk01 type= ServerDisk disk dsk02 type= ServerDisk
disk dsk03 type= ServerDisk;

server DBserver_WA type= IBM_RS/6000_M43P133
dbms= Oracle DB_BuffSize= 8192 num_CPUs= 2
disk dsk01 type= ServerDisk disk dsk02 type= ServerDisk
disk dsk03 type= ServerDisk;

Example of Clisspe Declaration Section for RTS System (continued)

! declaration of DB tables
table applicant num_rows= 1000000 row_size= 120 dbms= Oracle
columns= (ssn, name, city/200, zip/99999, education/10)
  index= (key= (city) key_size= 20 btree)
  index= (key= (zip) key_size= 5 btree clustered);

table skill num_rows= 200 row_size= 100 dbms= Oracle
columns= (SkillCode, SkillName, MinVal/4, MaxVal/4);

table course num_rows= 1000 row_size= 60 dbms= Oracle
columns= (coursenum, cname, hours/3)
  index= (key= (coursenum) key_size= 4 btree clustered);

table section num_rows= 10000 row_size= 70 dbms= Oracle
columns= (coursenum, SectionNum/10, StartDate/24, daytime/4,
  location/30, maxcap/4)
  index= (key= (location) key_size=20 btree clustered);
Example of Clisspe Declaration Section for RTS System (continued)

table enrollment num_rows= 400000 row_size= 20 dbms= oracle
  columns = (coursenum/1000, SectionNum/10, ssn)
  index= (key= (coursenum, SectionNum) btree clustered);

table ApplicantHasSkill num_rows= 5000000 row_size= 16 dbms= Oracle
  columns= (ssn/1000000, SkillCode/200, SkillValue/4)
  index= (key= (ssn) key_size= 9 btreeclustered);

Example of Clisspe Declaration Section for RTS System (continued)

table ApplicantNeedsSkill num_rows= 5000000 row_size= 16 dbms= Oracle
  columns= (ssn/1000000, SkillCode/200, SkillValue/4)
  index= (key= (ssn) key_size= 9 btree clustered);

table CourseDevelopsSkill num_rows= 3000 row_size= 12 dbms= Oracle
  columns= (coursenum/1000, SkillCode/200)
  index= (key= (coursenum) key_size= 4 btree clustered);
Example of Clisspe Declaration Section
for RTS System (continued)

! network declarations
network_type RecruitmentLan bandwidth= 10 type= Ethernet;
network_type CenterLan bandwidth= 100 type= Fast_Ethernet;
network_type Enterprise bandwidth= 45 type= WAN;

network LA_LAN type= RecruitmentLan;
network NY_LAN type= RecruitmentLan;
network WA_LAN type= RecruitmentLan;
network Center_Lan type= CenterLan;
network EnterpriseNet type= Enterprise;

transaction apply rate= 0.001;
transaction check_skills rate= 0.002;
transaction enroll rate= 0.0001;

! rpc declarations
rpc RPCtoApplServer
  local_time= 0.0015 benchmark= 30 (specint92)
  remote_time= 0.0030 benchmark= 40 (specint92)
  nbytes= 2048;

end_declaration;
Clisspe Mapping Section

- clients to networks
- servers to networks
- DB tables to servers
- transactions to clients
- network paths definitions
- transactions to network paths

Example of Clisspe Mapping Section for RTS System

mapping
! mapping of servers to networks
server appl_DC is_in network DC_LAN;
server appl_LA is_in network LA_LAN;
server appl_SE is_in network SE_LAN;
server DBServer is_in network CenterLan;
Example of Clisspe Mapping Section for RTS System (continued)

! mapping of clients to networks
client gc_DC is_in network DC_LAN;
client gc_LA is_in network LA_LAN;
client gc_SE is_in network SE_LAN;

Example of Clisspe Mapping Section for RTS System (continued)

! mapping of tables to servers
table applicant is_in server DBServer
  (dsk01: 0.3, dsk02: 0.3, dsk03: 0.4);
table skill is_in server DBServer
  (dsk01: 1.0);
table course is_in server DBServer
  (dsk02: 1.0);
table section is_in server DBServer
  (dsk01: 0.3, dsk02: 0.3, dsk03: 0.4);
Example of Clisspe Mapping Section for RTS System (continued)

table enrollment is in server DBServer
   (dsk01: 0.3, dsk02: 0.3, dsk03: 0.4);
table ApplicantHasSkill is in server DBServer
   (dsk01: 0.3, dsk02: 0.3, dsk03: 0.4);
table ApplicantNeedsSkill is in server DBServer
   (dsk01: 0.3, dsk02: 0.3, dsk03: 0.4);
table CourseDevelopsSkill is in server DBServer
   (dsk01: 0.3, dsk02: 0.3, dsk03: 0.4);

Example of Clisspe Mapping Section for RTS System (continued)

! mapping of transactions
transaction applyDC submitted by
   client gc_DC percent_rate= 0.2;
transaction applyLA submitted by
   client gc_LA percent_rate= 0.35;
transaction applySE submitted by
   client gc_SE percent_rate= 0.45;
Example of Clisspe Mapping Section for RTS System (continued)

\[
\text{transaction check\_skillsDC submitted\_by client gc\_DC percent\_rate} = 0.2; \\
\text{transaction check\_skillsLA submitted\_by client gc\_LA percent\_rate} = 0.35; \\
\text{transaction check\_skillsSE submitted\_by client gc\_SE percent\_rate} = 0.45;
\]

Example of Clisspe Mapping Section for RTS System (continued)

\[
\text{transaction enrollDC submitted\_by client gc\_DC percent\_rate} = 0.2; \\
\text{transaction enrollLA submitted\_by client gc\_LA percent\_rate} = 0.35; \\
\text{transaction enrollSE submitted\_by client gc\_SE percent\_rate} = 0.45;
\]
Example of Clisspe Mapping Section for RTS System (continued)

! network paths to application servers
net_path applDC from client gc_DC
to server appl_DC via networks DC_LAN;
net_path applLA from client gc_LA
to server appl_LA via networks LA_LAN;
net_path applSE from client gc_SE
to server appl_SE via networks SE_LAN;

Example of Clisspe Mapping Section for RTS System (continued)

! network paths from application to DB servers
net_path dbaccessDC from client gc_DC
to server appl_DC to server DBServer
via networks DC_LAN, EnterpriseNet, CenterLan;
net_path dbaccessLA from client gc_LA
to server appl_LA to server DBServer
via networks LA_LAN, EnterpriseNet, CenterLan;
net_path dbaccessSE from client gc_SE
to server appl_SE to server DBServer
via networks SE_LAN, EnterpriseNet, CenterLan;
Example of Clisspe Mapping Section for RTS System (continued)

! mapping of transactions to network paths
transaction applyDC uses net_path applDC routing_frequency= 1.0;
transaction applyLA uses net_path applLA routing_frequency= 1.0;
transaction applySE uses net_path applSE routing_frequency= 1.0;
transaction applyDC uses net_path dbaccessDC routing_frequency= 1.0;
transaction applyLA uses net_path dbaccessLA routing_frequency= 1.0;
transaction applySE uses net_path dbaccessSE routing_frequency= 1.0;

transaction check_skillsDC uses net_path applDC
    routing_frequency= 1.0;
transaction check_skillsLA uses net_path applLA
    routing_frequency= 1.0;
transaction check_skillsSE uses net_path applSE
    routing_frequency= 1.0;
transaction check_skillsDC uses net_path dbaccessDC
    routing_frequency= 1.0;
transaction check_skillsLA uses net_path dbaccessLA
    routing_frequency= 1.0;
transaction check_skillsSE uses net_path dbaccessSE
    routing_frequency= 1.0;
Example of Clisspe Mapping Section for RTS System (continued)

```plaintext
transaction enrollDC uses net_path applDC
    routing_frequency= 1.0;
transaction enrollLA uses net_path applLA
    routing_frequency= 1.0;
transaction enrollSE uses net_path applSE
    routing_frequency= 1.0;
transaction enrollDC uses net_path dbaccessDC
    routing_frequency= 1.0;
transaction enrollLA uses net_path dbaccessLA
    routing_frequency= 1.0;
transaction enrollSE uses net_path dbaccessSE
    routing_frequency= 1.0;
end_mapping;
```

Clisspe Transaction Section Statements

- DB query (select)
- DB update
- rpc
- compute
- if then else
- switch
- loop
Example of Clisspe Transaction Section for RTS System (continued)

! transaction apply
 transaction applyDC running_on client
  rpc RPCtoApplServer to_server appl_DC;
end_transaction;

transaction applyDC running_on server appl_DC
  ! check if applicant exists
  select from applicant where ssn;
  ! in ten percent of the cases the applicant is already in the DB
  if 0.9
  then ! add applicant to database
     update applicant num_rows= 1;
  end_if;
end_transaction; ! applyDC

Example of Clisspe Transaction Section for RTS System (continued)

! transaction check_skills
 transaction check_skillsDC running_on client
  ! check if applicant exists
  rpc RPCtoApplServer to_server appl_DC;
  ! if applicant exists check applicant skills
  if 0.9
  then rpc RPCtoApplServer to_server appl_DC;
   end_if;
end_transaction;
Example of Clisspe Transaction Section for RTS System (continued)

transaction check_skillsDC running_on server appl_DC
  ! check if applicant exists
  select from applicant where ssn;
  ! if applicant exists check applicant skills
  if 0.9
    then ! find all courses the applicant qualifies for
      select from ApplicantHasSkill where ssn
        from CourseRequiresSkill where CourseNum
          joined_by ApplicantHasSkill.SkillCode = CourseRequiresSkill.SkillCode;
    end_if;
  end_transaction;

Example of Clisspe Transaction Section for RTS System (continued)

transaction enrollDC running_on client
  ! for all courses to be enrolled
  loop #avg_courses_enrolled
    ! check seat availability for all sections
    loop #sections_checked
      rpc RPCtoApplServer to_server appl_DC;
      end_loop;
    ! enroll applicant in section
    rpc RPCtoApplServer to_server appl_DC;
  end_loop;
end_transaction;
Example of Clisspe Transaction Section for RTS System (continued)

transaction enrollDC running_on server appl_DC
  ! for all courses to be enrolled
  loop #avg_courses_enrolled
    ! check seat availability for all sections
    loop #sections_checked
      select from enrollment where coursenum;
      end_loop;
    ! enroll applicant in section
    update enrollment num_rows= 1;
  end_loop;
end_transaction;

Service Demand Generation

\[ D_{i,r} = \sum_{s \in S_{i,r}} n_s \times p_s \times D_{i,r}^s \]

where,

- \( D_{i,r}^s \): average service demand at device \( i \) for class \( r \) due to statement \( s \).
- \( S_{i,r} \): set of statements that generate demands at device \( i \) for class \( r \).
- \( n_s \): average number of times that statement \( s \) is executed.
- \( p_s \): probability that statement \( s \) is executed.
Service Demand Generation

if 0.3 then
    s1;
    s2;
else
    s3;
end_if

\[ p_{s1} = 0.3 \]
\[ p_{s2} = 0.3 \]
\[ p_{s3} = 0.7 \]

Service Demand Generation

switch
    case 0.1: s1;
    case 0.3: s2;
    case 0.6: s4;
end_switch;

\[ p_{s1} = 0.1 \]
\[ p_{s2} = 0.3 \]
\[ p_{s3} = 0.3 \]
\[ p_{s4} = 0.6 \]
Service Demand Generation

```plaintext
loop 3.5
    s1;
    s2;
    s3;
end_loop;
```

```
\[
\begin{array}{c}
    n_{s1} = 3.5 \\
    n_{s2} = 3.5 \\
    n_{s3} = 3.5 \\
\end{array}
\]
```

Service Demand Generation

```plaintext
loop 3.5
    if 0.3
    then s1;
        loop 2
            s2;
        end_loop;
    else s3;
    end_if;
end_loop;
```

```
\[
\begin{array}{cc}
    s & n_s & p_s \\
    s_1 & 3.5 & 0.3 \\
    s_2 & 7.0 & 0.3 \\
    s_3 & 3.5 & 0.7 \\
\end{array}
\]
```
Service Demand Generation

**Select**

appl server

```
select
```

network demand

db server

CPU and disk demands

**Update**

appl server

```
update
```

network demand

db server

CPU and disk demands

Service Demand Generation

**RPC**

client

```
client
```

network demand

 appl server

```
appl server
```

CPU demand

**RPC**

appl server

```
appl server
```

network demand

db server

CPU demand
Service Demand Generation

client or server

compute

CPU demand

Outline

• Motivating Example
• Basic Issues in SPE
• A Methodology for SPE in C/S
• Clisspe: A Language for SPE in C/S
• Benchmarking and Data Collection for SPE
• A Case Study
• Concluding Remarks
Benchmarking and Data Collection

- The CPU time associated with I/O is linear with number of bytes read/written.
- Simple benchmark programs can be written to obtain the parameters of the line
  \[ y = a x + b \]

The `server_type` command in Clisspe (see below) requires that the parameters \(a\) and \(b\) for the server be provided.

```plaintext
server_type SUNSparc20 specint92=150 specfp92=230
10_benchmark (a=0.2576, b=0.0082, specint92=100) ;
```
1. Create a file called IO_bench with at least 800 blocks of 512 bytes each.

2. Write a program called IO_time with the following pseudo code:

```c
#define Nblocks n /* the value of this constant will change for each 
                 execution of this program */
#define MAXBLOCKS 800 /* maximum number of blocks */
main ()
{
    int i, k, b;
    /* repeat experiment several times */
    for (k=1; k <= 50; ++k)
    { /* read n randomly selected blocks */
        for (i = 1; i <= n; ++i)
        { select an integer random number b between 1 and MAXBLOCKS;
            read block b from file IO_bench;
        }
    }
}
```

3. Write a program called Loop_time with the pseudo code given below. This program is identical to program IO_time except that it does not have the read statement inside the loop.

```c
#define Nblocks n /* the value of this constant will change for each 
                 execution of this program */
#define MAXBLOCKS 800 /* maximum number of blocks */
main ()
{
    int i, k, b;
    /* repeat experiment several times */
    for (k=1; k <= 50; ++k)
    { for (i = 1; i <= n; ++i)
        { select an integer random number b between 1 and MAXBLOCKS;
            read block b from file IO_bench;
        }
    }
```
Benchmarking and Data Collection

4. Run program IO_time for \( n = 1, 5, 10, 15, \) and 20. For each execution of the program obtain the CPU time (not the elapsed time) and divide it by 50.

5. Run program Loop_time for \( n = 1, 5, 10, 15, \) and 20. For each execution of the program obtain the CPU time (not the elapsed time) and divide it by 50.

6. Build the following table where the value in the last column is equal to the value in column 2 minus the value in column 3.

<table>
<thead>
<tr>
<th>N</th>
<th>CPU time for ( \text{IO time} / 50 )</th>
<th>CPU time for ( \text{Loop time} / 50 )</th>
<th>number of bytes (x)</th>
<th>CPU time for I/O in msec (y)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td>512</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td>2560</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td>5120</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td></td>
<td></td>
<td>7680</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td></td>
<td></td>
<td>10240</td>
<td></td>
</tr>
</tbody>
</table>

Apply linear regression to the last two columns of the table above to obtain the relationship 
\[ y = ax + b \]

Outline

- Motivating Example
- Basic Issues in SPE
- A Methodology for SPE in C/S
- Clisspe: A Language for SPE in C/S
- Benchmarking and Data Collection for SPE
- A Case Study
- Concluding Remarks

© 1997 Daniel A. Menascé
>>> List of Devices of the Performance Model

<table>
<thead>
<tr>
<th>Dev. No.</th>
<th>Name</th>
<th>#Servers</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>CL:gc_DC</td>
<td>1</td>
<td>Client</td>
</tr>
<tr>
<td>2</td>
<td>CL:gc_LA</td>
<td>1</td>
<td>Client</td>
</tr>
<tr>
<td>3</td>
<td>CL:gc_SE</td>
<td>1</td>
<td>Client</td>
</tr>
<tr>
<td>4</td>
<td>SP:appl_DC</td>
<td>1</td>
<td>Server Processor</td>
</tr>
<tr>
<td>5</td>
<td>SD:04:dsk01</td>
<td>1</td>
<td>Server Disk</td>
</tr>
<tr>
<td>6</td>
<td>SP:appl_LA</td>
<td>1</td>
<td>Server Processor</td>
</tr>
<tr>
<td>7</td>
<td>SD:06:dsk01</td>
<td>1</td>
<td>Server Disk</td>
</tr>
<tr>
<td>8</td>
<td>SP:appl_SE</td>
<td>1</td>
<td>Server Processor</td>
</tr>
<tr>
<td>9</td>
<td>SD:08:dsk01</td>
<td>1</td>
<td>Server Disk</td>
</tr>
<tr>
<td>10</td>
<td>SP:DBServer</td>
<td>2</td>
<td>Server Processor</td>
</tr>
<tr>
<td>11</td>
<td>SD:10:dsk01</td>
<td>1</td>
<td>Server Disk</td>
</tr>
<tr>
<td>12</td>
<td>SD:10:dsk02</td>
<td>1</td>
<td>Server Disk</td>
</tr>
<tr>
<td>13</td>
<td>SD:10:dsk03</td>
<td>1</td>
<td>Server Disk</td>
</tr>
<tr>
<td>14</td>
<td>NT:DC_LAN</td>
<td>1</td>
<td>Network</td>
</tr>
<tr>
<td>15</td>
<td>NT:LA_LAN</td>
<td>1</td>
<td>Network</td>
</tr>
<tr>
<td>16</td>
<td>NT:SE_LAN</td>
<td>1</td>
<td>Network</td>
</tr>
<tr>
<td>17</td>
<td>NT:CenterLan</td>
<td>1</td>
<td>Network</td>
</tr>
<tr>
<td>18</td>
<td>NT:EnterpriseNet</td>
<td>1</td>
<td>Network</td>
</tr>
</tbody>
</table>
Matrix of service demands (in sec):

```
>>> Matrix of service demands (in sec):

   1  2    3  4    5    ...  
1 0.00034 0.00000 0.00064 0.00000 0.00000  
2 0.00000 0.00034 0.00000 0.00000 0.00000  
3 0.00195 0.00000 0.00386 0.00000 0.00000  
4 0.00000 0.00000 0.00000 0.00000 0.00000  
5 0.00000 0.00000 0.00000 0.00000 0.00000  
6 0.00000 0.00000 0.00000 0.00000 0.00000  
7 0.00000 0.00000 0.00000 0.00000 0.00000  
8 0.00000 0.00000 0.00195 0.00000 0.00000  
9 0.00000 0.00000 0.00000 0.00000 0.00000  
10 0.00043 0.00043 0.00043 0.000097 0.000097  
11 0.02547 0.02547 0.02547 0.04098 0.04098  
12 0.02547 0.02547 0.02547 0.04098 0.04098  
13 0.03397 0.03397 0.03397 0.05464 0.05464  
14 0.00381 0.00000 0.00677 0.00000 0.00000  
15 0.00000 0.00000 0.00000 0.00000 0.00000  
16 0.00000 0.00000 0.00000 0.00000 0.00000  
17 0.00017 0.00017 0.00017 0.00031 0.00031  
18 0.00039 0.00039 0.00039 0.00068 0.00068  
```
### Bottleneck Analysis

<table>
<thead>
<tr>
<th>Transaction Name</th>
<th>Bottleneck % Contr.</th>
</tr>
</thead>
<tbody>
<tr>
<td>applyDC:gc_DC</td>
<td>SD:10:dsk03 37.02</td>
</tr>
<tr>
<td>applyLA:gc_LA</td>
<td>SD:10:dsk03 37.02</td>
</tr>
<tr>
<td>applySE:gc_SE</td>
<td>SD:10:dsk03 37.02</td>
</tr>
<tr>
<td>check_skillsDC:gc_DC</td>
<td>SD:10:dsk03 36.57</td>
</tr>
<tr>
<td>check_skillsLA:gc_LA</td>
<td>SD:10:dsk03 36.57</td>
</tr>
<tr>
<td>check_skillsSE:gc_SE</td>
<td>SD:10:dsk03 36.57</td>
</tr>
<tr>
<td>enrollDC:gc_DC</td>
<td>SD:10:dsk03 27.30</td>
</tr>
<tr>
<td>enrollLA:gc_LA</td>
<td>SD:10:dsk03 27.30</td>
</tr>
<tr>
<td>enrollSE:gc_SE</td>
<td>SD:10:dsk03 27.30</td>
</tr>
</tbody>
</table>

### Outline

- Motivating Example
- Basic Issues in SPE
- A Methodology for SPE in C/S
- *Clisspe*: A Language for SPE in C/S
- Benchmarking and Data Collection for SPE
- A Case Study
- Concluding Remarks
Concluding Remarks

- Software performance engineering for C/S systems requires a language to describe the clients, servers, networks, connectivity, DB tables, mappings, and transactions.
- The compiler for the language should generate a performance model that generates performance metrics for the C/S system.