Z_SOIL on 64-bit Windows system

Z_Soil Day 2009

Krzysztof Podleś
Windows 32 bit (x86) limitations

The primary limitation of 32-bit Windows operating system is the maximum allocatable system memory (RAM).

A single process on a 32-bit Windows operating system is limited to a total of 3.25 GB (gigabytes).

\[2^{32} = 4 \ 294 \ 967 \ 296\]
Windows 64 bit (x64) limitations

Theoretical memory limit a 64-bit computer can address is about 16 exabytes

\[2^{64} = 16 \times 1024 \times 1024 \times 1024 \times 1024 \times 1024 \times 1024\]

\[= 18\,446\,744\,073\,709\,551\,616\]
Windows XP (x64)

Windows XP x64 is limited to:
- 128 GB of physical memory
- 8 terabytes of virtual memory per process
Windows VISTA 64 bit (x64) limitations

• All 64-bit versions of Microsoft operating systems currently impose a **16 TB limit on address space.**
• Processes created on the 64-bit editions of Windows Vista can have 8 TB in virtual memory for user processes
• 8 TB for kernel processes to create a virtual memory of 16 TB.

In terms of physical memory
• Windows Vista 64-Bit Basic supports up to **8 GB of RAM**, Windows Vista 64-Bit Home Premium **16 GB of RAM**
• Windows Vista 64-Bit Business/Enterprise/Ultimate **128 GB of RAM.**
Windows 7

Maximum physical memory

• Home Basic 8 GB
• Home Premium 16 GB
• Professional 192 GB
• Enterprise 192 GB
• Ultimate 192 GB
# 32-bit data models

<table>
<thead>
<tr>
<th></th>
<th>short</th>
<th>int</th>
<th>long</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>16</td>
<td>32</td>
<td>32</td>
</tr>
<tr>
<td>range</td>
<td>–32,768</td>
<td>2,147,483,648</td>
<td>2,147,483,648</td>
</tr>
<tr>
<td></td>
<td>to 32,767</td>
<td>to 2,147,483,647</td>
<td>to 2,147,483,647</td>
</tr>
<tr>
<td></td>
<td>0 to 65,535</td>
<td>0 to 4,294,967,295</td>
<td>0 to 4,294,967,295</td>
</tr>
</tbody>
</table>
### 64-bit data models

<table>
<thead>
<tr>
<th>Data model</th>
<th>short</th>
<th>int</th>
<th>long</th>
<th>long long</th>
<th>pointers</th>
</tr>
</thead>
<tbody>
<tr>
<td>LLP64</td>
<td>16</td>
<td>32</td>
<td>32</td>
<td>64</td>
<td>64</td>
</tr>
<tr>
<td>LP64</td>
<td>16</td>
<td>32</td>
<td>64</td>
<td>64</td>
<td>64</td>
</tr>
<tr>
<td>ILP64</td>
<td>16</td>
<td>64</td>
<td>64</td>
<td>64</td>
<td>64</td>
</tr>
<tr>
<td>SILP64</td>
<td>64</td>
<td>64</td>
<td>64</td>
<td>64</td>
<td>64</td>
</tr>
</tbody>
</table>
64-bit data models in Z_Soil

• LLP64
  Preprocessing, Menu, Postprocessing
  Available total memory – as system
  Maximum array in one block – 16 GB

• LP64
  Calculation
  Available total memory – as system
  Maximum array in one block – 16 GB
New Compliers

- V 2009 – VC ++ 6.0
  Compaq Fortran 90 Compiler
- V 2010 Visual Studio 2008
- Intel Fortran Compiler
- VC ++ 11.0 Compiler – planned for testing
Tests

• Windows Vista Business
• 8 GB of physical memory
• 100 GB of virtual memory

• V 2009
• V 2010 (32-bit)
• V 2010 (64-bit)
Tests

For each examples one step of time dependent driver was calculated

Total time of the solution takes into account:
• Reading *.dat
• Aggregation of LHS and RHS
• Reordering
• Factorization
• Solution
• Saving results
Box

Box_55xx55x55.inp

55 elements

55 elements

55 elements
Box

60 elements

BOX_60xx60x60.inp
## Box – results

<table>
<thead>
<tr>
<th>Example</th>
<th>Nodes</th>
<th>Elements</th>
<th>Neq Ndofs</th>
<th>Total time [s]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>v2009</td>
</tr>
<tr>
<td>box 55x55x55</td>
<td>175 616</td>
<td>166 375</td>
<td>505 120</td>
<td>1498</td>
</tr>
<tr>
<td>box 60x60x60</td>
<td>226 981</td>
<td>216 000</td>
<td>655 140</td>
<td>x</td>
</tr>
</tbody>
</table>
### Box nonsymmetric matrix - results

<table>
<thead>
<tr>
<th>Example</th>
<th>Nodes</th>
<th>Elements</th>
<th>Ndofs</th>
<th>Neq</th>
<th>Total time [s]</th>
</tr>
</thead>
<tbody>
<tr>
<td>box 55x55x55</td>
<td>175 616</td>
<td>166 375</td>
<td>505 120</td>
<td>x</td>
<td>7154 3900</td>
</tr>
<tr>
<td>box 60x60x60</td>
<td>226 981</td>
<td>216 000</td>
<td>655 140</td>
<td>x</td>
<td>x 7469</td>
</tr>
</tbody>
</table>
Shell

400 elements

SH_400x400.inp
Shell

SH_800x800.inp
## Shell - results

<table>
<thead>
<tr>
<th>Example</th>
<th>Nodes</th>
<th>Elements</th>
<th>Ndofs Neq</th>
<th>Total time [s]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td><strong>v2009</strong></td>
</tr>
<tr>
<td>SH_400x400</td>
<td>160 801</td>
<td>160 000</td>
<td><strong>964 782</strong></td>
<td>470</td>
</tr>
<tr>
<td>SH_800x800</td>
<td>641 601</td>
<td>640 000</td>
<td><strong>3 849 582</strong></td>
<td>x</td>
</tr>
</tbody>
</table>
Building – shells + beams

11 storey

B_50x50x11.inp
Building – shells + beams

B_50x50x22.inp

50 elements

22 storey

50 elements
Building – shells + beams

B_50x50x44.inp

44 storey

B_50x50x88.inp

88 storey
## Building – results

<table>
<thead>
<tr>
<th>Example</th>
<th>Nodes</th>
<th>Elements</th>
<th>Ndofs Neq</th>
<th>Total time [s]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>v2009</td>
</tr>
<tr>
<td>sh_50x50x11</td>
<td>29 439</td>
<td>27 896</td>
<td>171 666</td>
<td>64</td>
</tr>
<tr>
<td>sh_50x50x22</td>
<td>58 842</td>
<td>55 792</td>
<td>343 332</td>
<td>127</td>
</tr>
<tr>
<td>sh_50x50x44</td>
<td>111 584</td>
<td>117 648</td>
<td>686 664</td>
<td>263</td>
</tr>
<tr>
<td>sh_50x50x44</td>
<td>223 168</td>
<td>235 296</td>
<td>1 373 328</td>
<td>x</td>
</tr>
</tbody>
</table>
Disk or memory

• IN CORE SOLVER
  8 GB is used - no way to work with other applications

• OUT OF CORE SOLVER
  User decide how much memory can be used for application. It is slower but it is possible to work with other application simultaneously
Fighting with bottlenecks

Bottleneck is a part of the code/program, where capacity of entire system is limited

It is not optimized functions/procedure
Split B8

- 50 x 50 x 50 - 180 s  reduced to  30 s
- 55 x 55 x 55 - 247 s  reduced to 43 s
- 60 x 60 x 60 - 399 s  reduced to 74 s

Split Q4/Shell one layer

- 400 x 400 - 6 hours  reduced to 15 s
- 800 x 800 - ~week   reduced to 43 s
Piles

User example with 1000 piles split by 15 and 80 000 elements
Save time in Preprocessing was reduced from 17 min to 12 s

Virtual to real
Create 100 000 elements from virtual mesh to real was reduced by factor 100