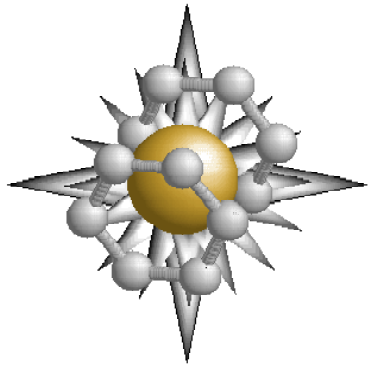


To Transpose or Not to Transpose; That is the Question

Performance Issues in the use of Java-based XML Software for Real-World Problems

Herbert J. Bernstein
Bernstein + Sons, Bellport, NY

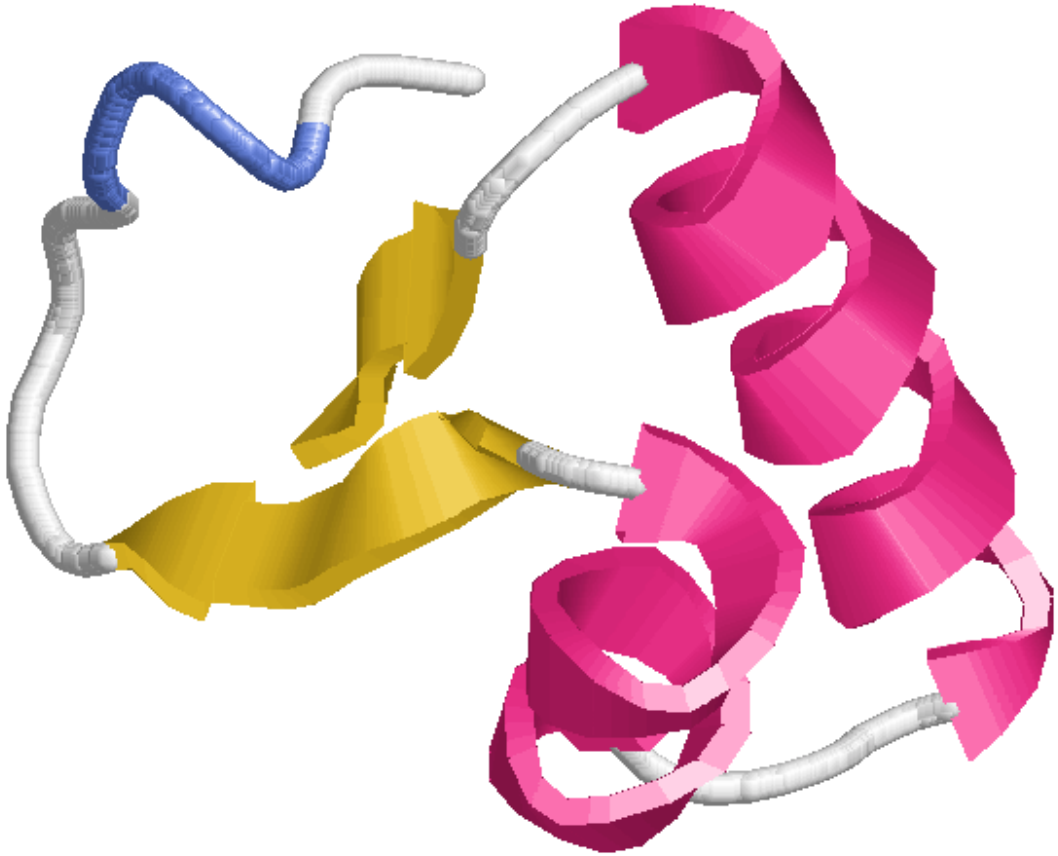
Design and Analysis Research Seminar
Computer Science Department, SUNY at Stony Brook
Wed., Sept. 26, 2001
2-3pm, CS Seminar Room 1306

Work supported in part by NSF via NDB project at Rutgers

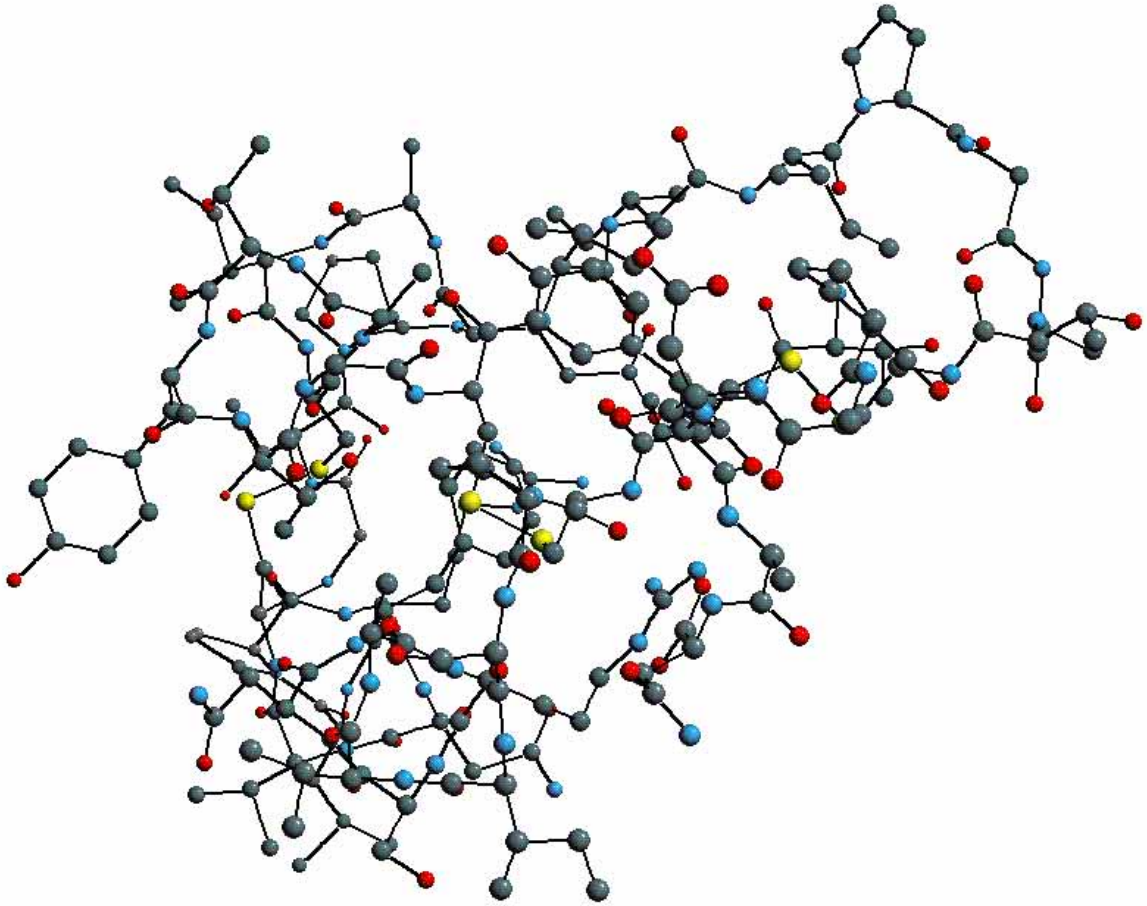
Preface

- Started in Bioinformatics
  - Managing large structural data sets
  - Drawing representations of molecules
- Older representations were fixed field, order dependent
  - Natural for arrays and procedural languages
- Newer representations are free field, order independent
  - Natural for trees and object-oriented languages
- Experience revealed serious performance issues
  - Apply to any language with dynamic memory allocation
  - Current impact is in high performance graphics
  - Likely to be a general issue as use of XML for large complex documents spreads

Typical drawing from fixed field file (1CRN)



Drawing from XML version of 1CRN using Jmol



Introduction

- The question: Java performance for large XML documents
- How the question arose
  - Data representation in Bioinformatics
  - PDB format, CIF, CML and XML
- XML is seeing more use
  - Document publication markup
  - Tree-oriented data representation
  - XHTML
- Java is often used to parse XML documents
- Java memory management
  - Highly dynamic
  - Aggressive garbage collection
- The problem
  - Exponential slowdowns
- Options to explore
  - Improvements to memory management
  - Control of garbage collection
  - Transposition of order of tree building
  - Build arrays instead of trees
- Status
  - Transposition works
  - Arrays need to be explored

The question:
Java performance for large XML documents

Despite recent improvements in Java VMs, there is a memory-management related performance problem when using Java to parse large tree-based documents.

- **See Ian Kaplan, "Memory Allocation and Garbage Collection," http://www.bearcave.com/software/garbage.htm, September 2000**

> **"Poor performance. The openGL graphics library supports "display lists" which describe the 3D polygons that make up a shape. A complex scene may have a large number of display lists. Deallocation of a display list, one element at a time, can be very time consuming and can have a large impact on program performance. This can be avoided using a block based memory allocator, where the display list elements are allocated from large memory blocks. When a display list is no longer needed, a pool of blocks can be deallocated. While this technique is effective, designing, implementing and debugging a good block allocator is time consuming."**
>
> **[Note: OpenGL is not part of Java, but this gives a strong clue to the problem, and a possible solution. -- HJB]**

- **Confirmed by our own experience parsing CML documents**
- **Confirmed in independent experiments at Rutgers (by J. Westbrook)**

How the question arose

- **Data representation in Bioinformatics**
  - **Very large data sets of atomic coordinates**
  - **Very large number of data sets**

- **PDB format, CIF, CML and XML**
  - **Old PDB format -- fixed field, tabular, order dependent**
  - **CIF -- tagged free field, tabular with normalization, not order-dependent**
  - **CML and XML -- tagged free field, tree-oriented, not order dependent**

Old PDB Format

The Protein Data Bank format [Bernstein, F.C., Koetzle, T.F., Williams, G. J. B., Meyer Jr., E. F., Brice, M.D.,Rodgers, J. R.,Kennard, O.,Shimanouchi, T., and Tasumi, M., "The Protein Data Bank: A Computer-based Archival File for Macromolecular Structures", J. Mol. Biol., 112, 535-542 (1977)].

```
ATOM      1  N   THR     1      17.047  14.099   3.625  1.00 13.79      1CRN 70
ATOM      2  CA  THR     1      16.967  12.784   4.338  1.00 10.80      1CRN 71
ATOM      3  C   THR     1      15.685  12.755   5.133  1.00  9.19      1CRN 72
ATOM      4  O   THR     1      15.268  13.825   5.594  1.00  9.85      1CRN 73
ATOM      5  CB  THR     1      18.170  12.703   5.337  1.00 13.02      1CRN 74
ATOM      6  OG1 THR     1      19.334  12.829   4.463  1.00 15.06      1CRN 75
```

- **Used for over 20 years to archive macromolecular data**
- **Simple, fixed field, tabular, ordered representation**
- **Suitable for procedural languages**
- **Does not need dynamic memory management.**
- **Highly efficient processing and graphics programs exist**
  In Fortran, C, C++, perl

Crystallographic Information File (CIF)

Simple Tag-Value Language (STAR)

S. R. Hall, "The STAR File: A New Format for Electronic Data Transfer and Archiving", *J. Chem. Inform. Comp. Sci.,* Vol. 31, 1991, pp. 326-333

Plus Extensible Dictionaries (core CIF, mmCIF, pdCIF)

```
_cell_length_a                    13.128(6)
_cell_length_b                    22.438(15)
_cell_length_c                    23.068(9)
_cell_angle_alpha                 89.57(7)
_cell_angle_beta                  91.08(6)
_cell_angle_gamma                 119.53(7)
```

```
loop_
_struct_sheet.id
_struct_sheet.number_strands
  A     3
  B    11
```

S. R. Hall, F. H. Allen and I. D. Brown, "The Crystallographic Information File (CIF): A New Standard Archive File for Crystallography", *Acta Cryst.*, Vol. A47, 1991, pp. 655-685.

Crystallographic Information File (CIF) (cont.)

- Simple tag-value structure
- Minimal syntactic noise
- Free field, order-independent, strongly tabular
- Suitable for both procedural and object-oriented languages
  - Handled with arrays, lists and trees
  - Best handled with dynamic memory allocation
- Two major styles
  - Small molecules -- handled without normalization
  - Macromolecules -- complex table linkages and normalization

This is a partial example of a small molecule coordinate list [Longridge 98]:

```
loop_
   _atom_site_label
   _atom_site_fract_x
   _atom_site_fract_y
   _atom_site_fract_z
   _atom_site_U_iso_or_equiv
   _atom_site_adp_type
   _atom_site_calc_flag
   _atom_site_refinement_flags
   _atom_site_occupancy
   _atom_site_disorder_assembly
   _atom_site_disorder_group
   _atom_site_type_symbol
   Fe1 1 0 1  .0084(2) Uani d S 1 . . Fe
   Na1  .50907(11)  .13980(8) 1.09450(9)  .0185(3) Uani d . 1 . . Na
   Na2  .89904(10)  .37128(8) 1.21657(9)  .0171(3) Uani d . 1 . . Na
   C1  .7997(2)  -.01740(18) 1.0419(2)  .0110(4) Uani d . 1 . . C
   N1  .6788(2)  -.02885(18) 1.0696(2)  .0166(4) Uani d . 1 . . N
   C2  .9306(3)  -.01004(16)  .8075(3)  .0130(4) Uani d . 1 . . C
   N2  .8896(2)  -.01832(19)  .6897(2)  .0180(5) Uani d . 1 . . N
   C3  .9777(2)  .1677(2)  .99877(17)  .0124(4) Uani d . 1 . . C
   N3  .9687(3)  .26941(19) 1.00269(19)  .0202(4) Uani d . 1 . . N
   O1  .8170(2)  .23149(18)  .6226(2)  .0353(5) Uani d D 1 . . O
  ...
```

This is an example of a macromolecular CIF (1CRN) as converted to mmCIF by the program pdb2cif [Bernstein et al. 98]

```
loop_
_atom_site.label_seq_id
_atom_site.group_PDB
_atom_site.type_symbol
_atom_site.label_atom_id
_atom_site.label_comp_id
_atom_site.label_asym_id
_atom_site.auth_seq_id
_atom_site.label_alt_id
_atom_site.cartn_x
_atom_site.cartn_y
_atom_site.cartn_z
_atom_site.occupancy
_atom_site.B_iso_or_equiv
_atom_site.footnote_id
_atom_site.label_entity_id
_atom_site.id
1
 ATOM  N  N    THR *    1  .  17.047  14.099   3.625  1.00 13.79  .    1     1
1
 ATOM  C  CA   THR *    1  .  16.967  12.784   4.338  1.00 10.80  .    1     2
1
 ATOM  C  C    THR *    1  .  15.685  12.755   5.133  1.00  9.19  .    1     3
1
 ...
```

Note that in both cases the coordinates are presented by rows, but columns are easily extracted.

XML

The full definition of XML is given in [Bray, Paoli, Sperberg-McQueen 98]. An XML document consists of

- **character data intermingled with "markup".**
- **ampersand ("&"), percent ("%"), angle brackets ("<" and ">") significant.**

- **Markup brackets marked-up text between start tags and end tags**

- **Free field, tree-oriented, not necessarily order dependent**

- **Molecules represented in XML using Chemical Markup Language (CML)**
  - **CML data may be column-oriented**
  - **CML data may be row-oriented**
  - **Reliable column extraction requires building a full tree**

Chemical Markup Language (CML)

XML has been used as a framework for definition of a Chemical Markup Language (CML) [Murray-Rust, Rzepa 99]. Here is an elided portion of the atom list from 1CRN as a column-oriented CML document as created by cif2xml [Bernstein, Bernstein 00]

```
<atomArray>
<atom_site.label_seq_id> 1 1 1 1 ... 46 46 46 46
</atom_site.label_seq_id>
<atom_site.group_PDB> ATOM ATOM ATOM ATOM ... ATOM ATOM ATOM ATOM
</atom_site.group_PDB>
<stringArray builtin="elementType"> N C C O ... C O N O </stringArray>
<stringArray title="atomName"> N CA C O ... CG OD1 ND2 OXT </stringArray>
<stringArray title="atomresName"> THR THR THR THR ... ASN ASN ASN ASN
</stringArray>
<stringArray title="chain"> * * * * ... * * * * </stringArray>
<atom_site.auth_seq_id> 1 1 1 1 ... 46 46 46 46 </atom_site.auth_seq_id>
<atom_site.label_alt_id> . . . . ... . . . . </atom_site.label_alt_id>
<floatArray builtin="x3"> 17.047 16.967 15.685 15.268 ... 12.538 11.982 13.407 12.703
</floatArray>
<floatArray builtin="y3"> 14.099 12.784 12.755 13.825 ... 4.769 4.304 4.849 3.298 4.973
</floatArray>
<floatArray builtin="z3"> 3.625 4.338 5.133 5.594 ... 14.922 15.886 15.015 10.746
</floatArray>
<floatArray builtin="occupancy"> 1. 1. 1. 1. ... 1. 1. 1. 1. </floatArray>
<floatArray title="B" convention="PDB"> 13.79 10.8 9.19 9.85 ... 7.98 11. 10.32 7.86
</floatArray>
<atom_site.footnote_id> . . . . ... . . . . </atom_site.footnote_id>
<atom_site.label_entity_id> 1 1 1 1 ... 1 1 1 1 </atom_site.label_entity_id>
<stringArray builtin="atomId"> 1 2 3 4 ...  324 325 326 327 </stringArray>
</atomArray>
```

Chemical Markup Language (CML) (cont.)

Alternatively, a row-oriented presentation could be used, as in the following example of the coordinates of 1CRN, as translated by cif2xml [Bernstein, Bernstein 00]

```
<atomArray>
<atom_site.label_seq_id>
1
</atom_site.label_seq_id>
<atom_site.group_PDB>
  ATOM
</atom_site.group_PDB>
<stringArray builtin="elementType">
       N
</stringArray>
<stringArray title="atomName">
         N
</stringArray>
<stringArray title="atomresName">
            THR
</stringArray>
<stringArray title="chain">
               *
</stringArray>
<atom_site.auth_seq_id>  1
</atom_site.auth_seq_id>
<atom_site.label_alt_id>    .
</atom_site.label_alt_id>
<floatArray builtin="x3">     17.047
</floatArray>
<floatArray builtin="y3">           14.099
</floatArray>
<floatArray builtin="z3">                  3.625
</floatArray>
<floatArray builtin="occupancy">
1.00
</floatArray>
<floatArray title="B" convention="PDB">
13.79
</floatArray>
<atom_site.footnote_id>
.
</atom_site.footnote_id>
<atom_site.label_entity_id>
1
</atom_site.label_entity_id>
<stringArray builtin="atomId">
1
</stringArray>
<atom_site.label_seq_id>
1
</atom_site.label_seq_id>
<atom_site.group_PDB>
  ATOM
</atom_site.group_PDB>
<stringArray builtin="elementType">
       C
</stringArray>
<stringArray title="atomName">
         CA
```

XML is seeing more use

- **Document publication markup**
- **Tree-oriented data representation**
- **XHTML**

Java, with extensive browser support, is a very natural choice for handling XML documents.

Performance issues now seen in graphical and macromolecular applications will have impact on handling large documents in other domains of Java application.

Java is often used to parse XML documents

- **DOM -- parse XML to trees and groves**
- **SAX -- event-driven parse**

DOM

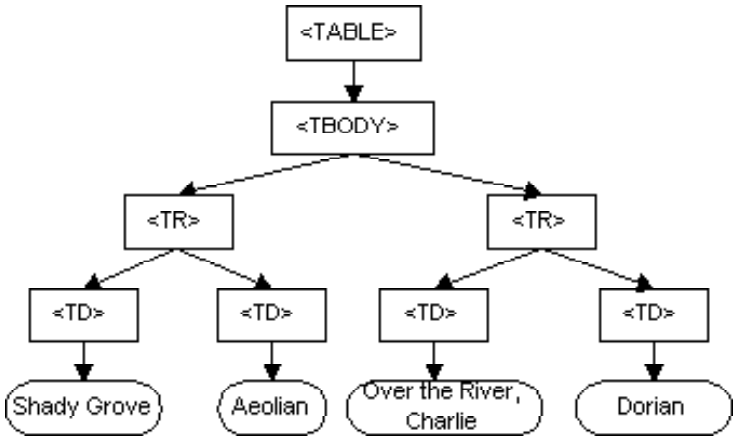
(from "What is the Document Object Model?", 13 Nov 2000, http://www.w3.org/TR/DOM-Level-2-Core/introduction.html, Editors: Philippe Le Hégaret, W3C, Lauren Wood, SoftQuad Software Inc., WG Chair, Jonathan Robie, Texcel (for DOM Level 1))

What the Document Object Model is

The DOM is a programming API for documents. It is based on an object structure that closely resembles the structure of the documents it models. For instance, consider this table, taken from an HTML document:

```
<TABLE>
<TBODY>
<TR>
<TD>Shady Grove</TD>
<TD>Aeolian</TD>
</TR>
<TR>
<TD>Over the River, Charlie</TD>
<TD>Dorian</TD>
</TR>
</TBODY>
</TABLE>
```

A graphical representation of the DOM of the example table is:



graphical representation of the DOM of the example table

"In the DOM, documents have a logical structure which is very much like a tree; to be more precise, which is like a "forest" or "grove", which can contain more than one tree. ..."

SAX

(from "SAX 2.0: The Simple API for XML", 5 May 2000, http://www.megginson.com/SAX/index.html, David Megginson, Megginson Technologies)

"SAX 2.0: ... API for event-based XML parsing.

> **"SAX2 ... new version of the popular Simple API for XML, incorporating support for Namespaces, for filter chains, and for querying and setting features and properties in the parser. ... .**

"SAX, the *Simple API for XML*, is a standard interface for event-based XML parsing, developed collaboratively by the members of the XML-DEV mailing list, currently hosted by OASIS. ... "

Java Memory Management

- **Highly dynamic**
  Most data structures allocated dynamically from the heap

- **Aggressive garbage collection**
  Most implementations search reference trees

Java Memory Management (cont.)

(from "Java Developer ConnectionSM (JDC) Tech Tips", December 22, 2000, http://developer.java.sun.com/developer/TechTips/2000/tt1222.html, by Stuart Halloway)

"A MEMORY TESTBED APPLICATION

> "Memory management can have a dramatic effect on performance, and most virtual machines expose a set of configuration options that you can tweak for the best possible performance of your application on a particular platform."

```
import java.io.*;
import java.util.*;

public class MemWorkout {
  private static final int K = 1024;
  private int maxStep;
  private LinkedList blobs = new LinkedList();
  private long totalAllocs;
  private long totalUnrefs;
  private long unrefs;

  public String toString() {
    return "MemWorkout allocs=" + totalAllocs +
" unrefs=" + totalUnrefs;
  }

  private static class Blob {
    public final int size;
    private final byte[] data;
    public Blob(int size) {
      this.size = size;
      data = new byte[size];
    }
  }

  private void grow(long goal) {
    long totalGrowth = 0;
    long allocs = 0;
    while (totalGrowth < goal) {
      int grow = (int)(Math.random() * maxStep);
      blobs.add(new Blob(grow));
      allocs++;
      totalGrowth += grow;
    }
    totalAllocs += allocs;
    System.out.println("" + allocs + " allocs, " +
totalGrowth + " bytes");
  }

  private void shrink(long goal) {
    long totalShrink = 0;
    unrefs = 0;
```

```
    try {
      while (totalShrink < goal) {
        totalShrink += shrinkNext();
      }
    } catch (NoSuchElementException nsee) {
      System.out.println("all items removed");
    }
    totalUnrefs+= unrefs;
    System.out.println("" + unrefs +
" unreferenced objs, " + totalShrink + " bytes");
  }

  private long shrinkNext() {
    //choice of FIFO/LIFO very important!
    Blob b = (Blob) blobs.removeFirst();
    //Blob b = (Blob) blobs.removeLast();
    unrefs++;
    return b.size;
  }

  public MemWorkout(int maxStep) {
    this.maxStep = maxStep;
  }

  public static void main(String [] args) {
    if (args.length < 1) {
      throw new Error (
"usage MemWorkout maxStepKB");
    }
    int maxStep = Integer.parseInt(args[0]) * K;
    if (maxStep < (K)) throw new Error(
"maxStep must be at least 1KB");
    MemWorkout mw = new MemWorkout(maxStep);
    try {
      while (true) {
        BufferedReader br = new BufferedReader(new
              InputStreamReader(System.in));
        logMemStats();
        System.out.println("{intMB} allocates, {-intMB}
deallocates, GC collects garbage, EXIT exits");
        String s = br.readLine();
        if (s.equals("GC")) {
          System.gc();
          System.runFinalization();
          continue;
        }
        long alloc = Integer.parseInt(s) * 1024* 1024;
        if (alloc > 0) {
          mw.grow(alloc);
        } else {
          mw.shrink(-alloc);
        }
      }
    } catch (NumberFormatException ne) {
    } catch (Throwable t) {
      t.printStackTrace();
    }
    System.out.println(mw);
  }

  public static void logMemStats() {
    Runtime rt = Runtime.getRuntime();
    System.out.println("total mem: " +
              (rt.totalMemory()/K) + "K free mem:  " +
(rt.freeMemory()/K) + "K");
  }
}
```

Java Memory Management (cont.)

(from "Java Developer ConnectionSM (JDC) Tech Tips", December 22, 2000, http://developer.java.sun.com/developer/TechTips/2000/tt1222.html, by Stuart Halloway)

"CONTROLLING YOUR MEMORY MANAGER

"Garbage collection performance can be very important to the overall performance of an application written in the Java programming language. The most primitive memory management schemes use a "stop-the-world" approach, where all other activity in the VM must halt while all objects in the system are scanned. This can cause a noticeable pause in program execution. Even when delays do not come in large, user-irritating chunks, the overall time spent collecting garbage can still impact performance. ...:

Flags	Purpose
-Xms and -Xmx	Control system memory usage
-verbose:gc	Trace garbage collection activity
-XX:NewSize	Control the nursery starting size
-Xincgc and -Xnoincgc	Turn on, or off, incremental garbage collection

"***Warning*** The -X flags are non-standard options, and are subject to change in future releases of the Java 2 SDK. Also, the -XX flag is officially unsupported, and subject to change without notice."

Useful Garbage Collection Information

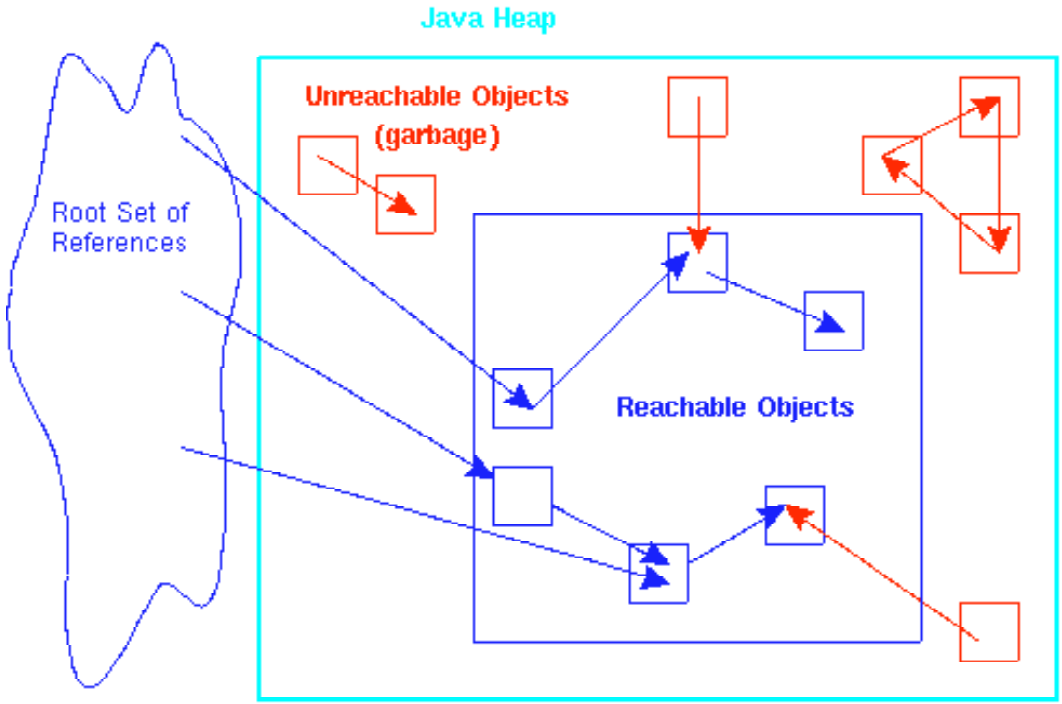
"Java's garbage-collected heap: An introduction to the garbage-collected heap of the Java virtual machine," by Bill Venners, http://www.javaworld.com/javaworld/jw-08-1996/jw-08-gc.html, Javaworld, August 1996,

"*the* Garbage Collection Page," by Richard Jones, http://www.cs.ukc.ac.uk/people/staff/rej/gc.html, 11 January 2001,

"Reference Objects and Garbage Collection," by Monica Pawlan, http://developer.java.sun.com/developer/technicalArticles/ALT/RefObj/, Java Developer Connection, August 1998.

Useful Garbage Collection Information (Cont.)

(from "Reference Objects and Garbage Collection," by Monica Pawlan, http://developer.java.sun.com/developer/technicalArticles/ALT/RefObj/, Java Developer Connection, August 1998.



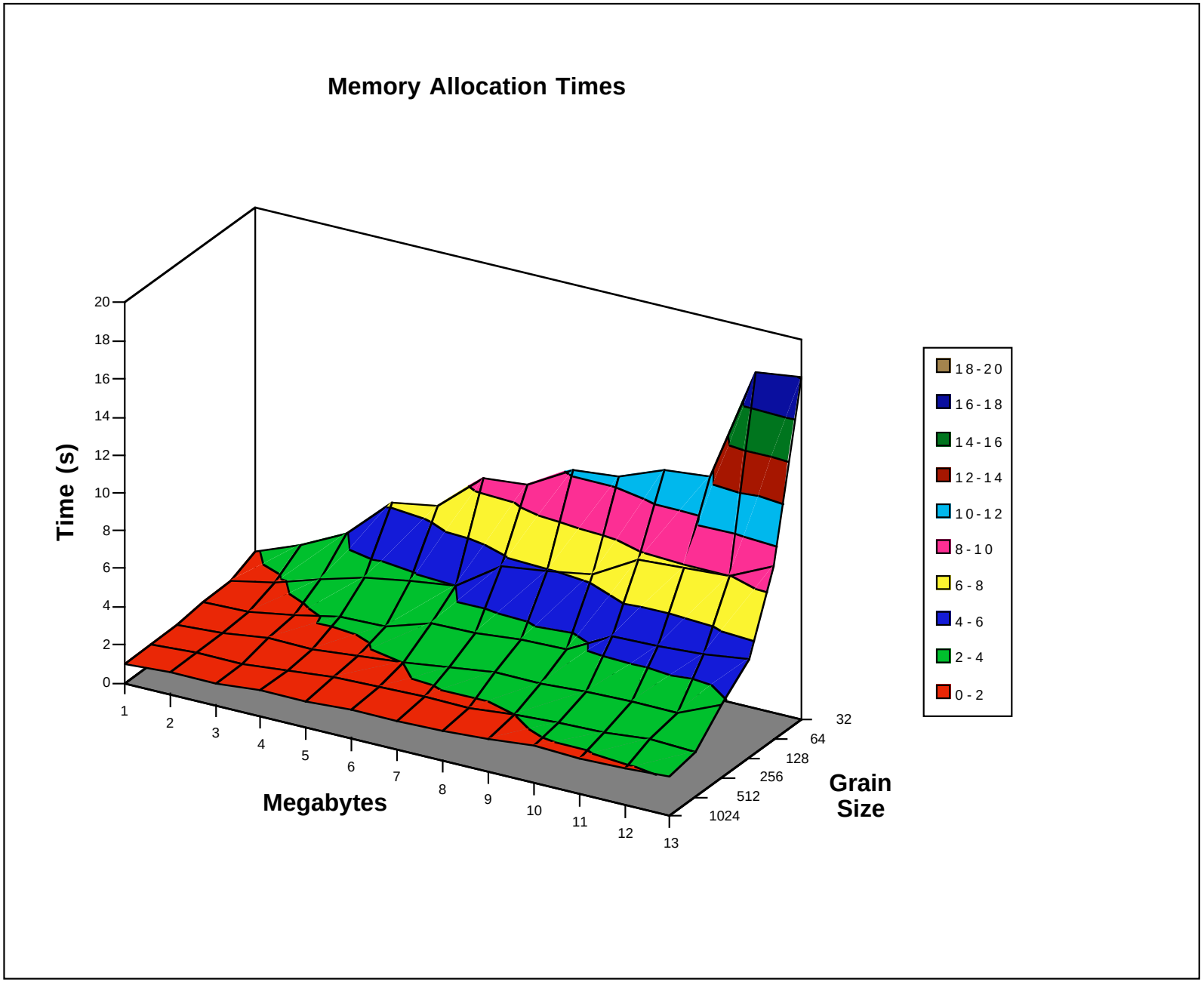
The Problem

- **Exponential slowdowns**
  - **Encountered when trying Jmol on macromolecules**
  - **Verified independently at Rutgers**
  - **Explored with modified form of MemWorkout**

Jmol: The program Jmol [Gezelter 99]

- **Java program able to display CML datasets.**
- **Performance on small molecules is excellent.**
- **When tried on large row-oriented macromolecules, molecule loads would never complete.**
  - **Column oriented loads worked, slowly but effectively.**
- **Data format verified:**
  - **Output of cif2xml by rows agrees with output of BioDOM program pdb2xml [Moore 99]**
- **For coordinate lists the higher information density of output by columns results in faster dataset reading.**

Row oriented problem partly due to syntactic noise of redundant XML tags, but experience with CIF, which has less syntactic noise, points to a problem with parses to many small-grained memory allocations. Verified with a modified version of MemWorkout:


Memory Allocation Times
Time (s)
20
18
16
14
12
10
8
6
4
2
0
1
2
3
4
5
6
7
8
9
10
11
12
13
Megabytes
32
64
128
256
512
1024
Grain Size
18-20
16-18
14-16
12-14
10-12
8-10
6-8
4-6
2-4
0-2

Options to explore

- **Improvements to memory management**
  - **Move to Java 1.3 did improve performance from unacceptable to less unacceptable**
- **Control of garbage collection**
  - **In building large trees, most of what is allocated is not available for release**
- **Transposition of order of tree building**
  - **Allocating in large chunks is effective**
  - **Allocate large columns instead of short rows**
- **Build arrays instead of trees**
  - **Achieve linear time by allocating geometric progression of larger arrays and copying**
  - **No overhead for tree links**

Status

- **Transposition works**

  Jmol is usable with column-oriented data and moderate macromolecule sizes

- **Arrays need to be explored**

  For the largest molecules load times are unacceptable

  Similar changes are needed for C, C++-based CIF parsers

Allocation in Expanding Blocks

(suggested by D. Bernstein, UIC)

- **Build tables in pre-allocated block of size X**
- **When block fills, allocate a new block of size αX, $\alpha > 1$**
  - **Copy old data from old block to new**
  - **Release the old block (explicitly or implicitly)**
- **If the time to allocate a block of size Y is βY, i.e. linear in Y (ignoring setup time) and the time to copy a block of size Z is γZ, i.e. linear in Z (ignoring setup time), then the times to set up various block sizes are:**

Size	Time
X	βX
αX	$\beta X+\alpha\beta X+\gamma X$
$\alpha^2 X$	$\beta X+\alpha\beta X+\gamma X + \alpha^2\beta X + \gamma\alpha X$
$\alpha^3 X$	$\beta X+\alpha\beta X+\gamma X + \alpha^2\beta X + \gamma\alpha X + \alpha^3\beta X + \gamma\alpha^2 X$
...	
$\alpha^n X$	$((\alpha^{(n+1)}-1)\beta X + (\alpha^n-1)\gamma X)/(\alpha-1)$

i.e. approximately linear in the block size.

References

[Bernstein, Bernstein 00] Bernstein, H.J.,Bernstein, F.C. "xmlCIF: A Proposal for Faithful Translation between XML and Extended CIF," poster at St. Paul ACA Meeting, 22-27 July 2000, Abstract E0009.

[Bernstein et al. 98] Bernstein, H.J.,Bernstein, F.C., Bourne, P.E. "pdb2cif: Translating PDB Entries into mmCIF Format," , J. Appl. Cryst., 31, pp. 282-295, 1998, software available from http://www.iucr.org/iucr-top/CIF and http://ndbserver.rutgers.edu.

[Bray, Paoli, Sperberg-McQueen 98] Bray, T., Paoli, J., Sperberg, C. M., eds, "Extensible Markup Language (XML)", W3C Recommendation 10-Feb-98, REC-xml-19980210, http://www.w3.org/TR/1998/REC-xml-19980210

[Fitzgerald et al. 96] Fitzgerald, P. M. D., Berman, H. M., Bourne, P. E., McMahon, B., Watenpaugh, K., Westbrook, J. "The mmCIF Dictionary: Community Review and Final Approval," 17th IUCR Congress and General Assembly, Seattle, Washington, USA, 8-17 August 1996, Abstract E1226. Version 0.8.02 available from http://ndbserver.rutgers.edu.

[Gezelter 99] Gezelter, D., “Jmol” an open source Java program. See http://www.openscience.org/jmol.

[Hall, Allen, Brown 91] Hall,S. R. Allen, F. H., Brown, I. D., "The Crystallographic Information File (CIF): A New Standard Archive File for Crystallography", Acta Cryst. A47, 655-685 (1991), http://www.us.iucr.org/iucr-top/cif/standard/cifstd1.html

[Hall, Bernstein 96] Hall, S.R., Bernstein, H.J., "CIFtbx2: Extended Tool Box for Manipulating CIFs," J. Appl. Cryst., 29, pp 598-603 (1996).

[Hendrickson, Teeter 81] Hendrickson, W. A., Teeter, M. M., “Crambin”, PDB Entry 1CRN. See also Teeter, M. M., “Water Structure Of A Hydrophobic Protein At Atomic Resolution. Pentagon Rings Of Water Molecules In Crystals Of Crambin”, Proc. Nat. Acad. Sci., USA, 81, 6014 ff. (1984).

[Longridge 98] Longridge, J. J., "Tetrasodium Hexacyanoferrate(II) Decahydrate", Acta Cryst. C54, 1998, CIF-Access paper, IUCR9800028.cif.

[Moore 99] Moore, A., “pdb2xml”, March 1999, released as pdb2xml-protbot.pl on the BioDOM website, http://ala.vsms.nottingham.ac.uk/biodom/software/protsuite-user-dist/

[Murray-Rust, Rzepa 99] Murray-Rust, P., Rzepa, H., "Chemical markup, XML and the WWW, Part I: Basic principles," J. Chem. Inf . Comp. Sci, 39 No. 6, 928-942,(1999). See http://www.xml-cml.org.