

Grids 1.0 beta and beyond

Andy Turner

<http://www.geog.leeds.ac.uk/people/a.turner/>



Outline

- Introduction
- Detail
 - What
 - Why
 - Memory Handling
- Next Steps
- Summary
- Questions Comments Advice



Introduction



Who are we?

- Andy Turner
 - Researcher
 - Computational geographer
 - Java programmer
 - e-Social Science in action!
- You
 - Similar?
 - Who else?



What is Grids 1.0 beta in a nutshell?

- Java for processing numeric 2D Square Celled raster data
- Open Source LGPL research software
- Beta 1 released in March 2005
- Beta 6 released in March 2006 (latest)
- Releases available via

<http://www.geog.leeds.ac.uk/people/a.turner/src/java/grids>



Why develop Grids?



Original Motivation

- Learn Java
- Other software did not really do what I wanted
- Develop a generally useful technology to support applications
- To control
 - Numerical accuracy (precision)
 - Error handling
- To build a component on which other software can be based
 - Geographically Weighted Statistics (GWS) etc...



What couldn't other software do?

- Handle a single raster data layer with hundreds of thousands of rows and columns
- That was in the year 2000



Maybe some (your?) software
could... but anyway...



...much of this Original Motivation
is still reason for developing
Grids...



- Java is evolving
- Data sets get larger
- I still don't know of any software that can do the things I want
- I still want control over numbers and errors
- I am still developing other Java based on Grids



The state of Grids

- 1.0 because
 - API is feature complete (relatively stable)
 - It has to reach 1.0 at some stage!
- Beta because
 - Documentation is OK but not great
 - No unit tests
 - Not enough good examples
- Used in teaching
 - A lecture, practical and workshop on a GIS and Environment module at the University of Leeds
 - Run annually
 - Mixed bag of students are a great help



More detailed
description...



Package Structure

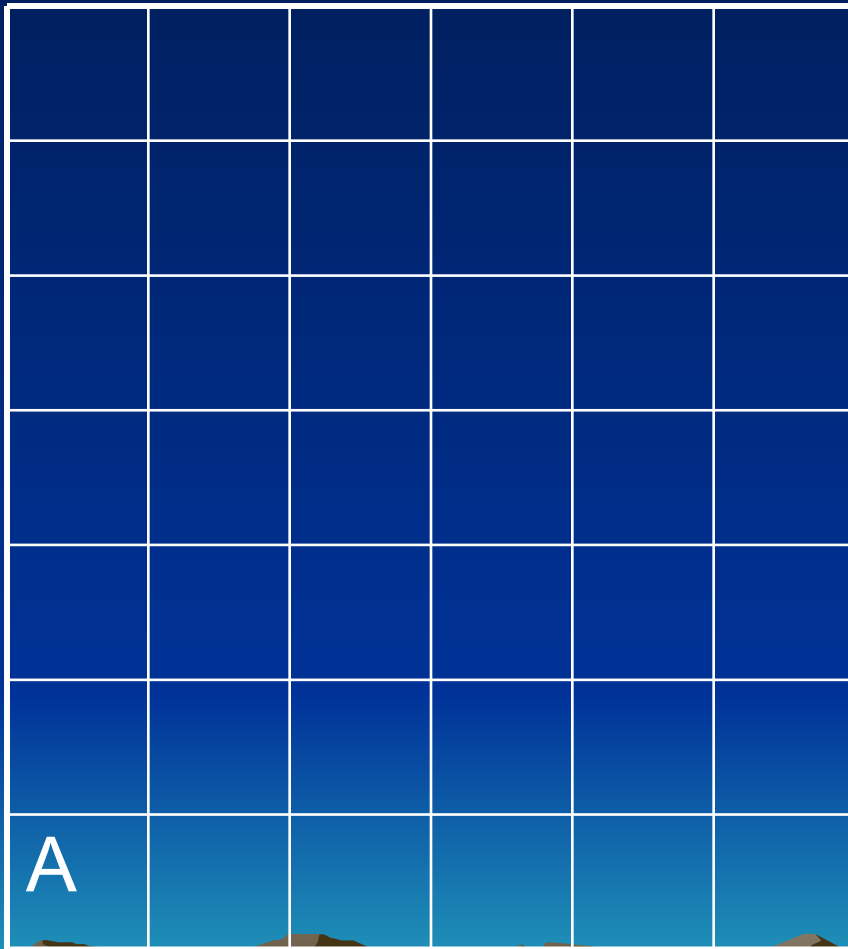
- core
- process
 - Sets of methods for particular kinds of processing
- utilities
 - Generic code
- exchange
 - For loading and saving Grids



core



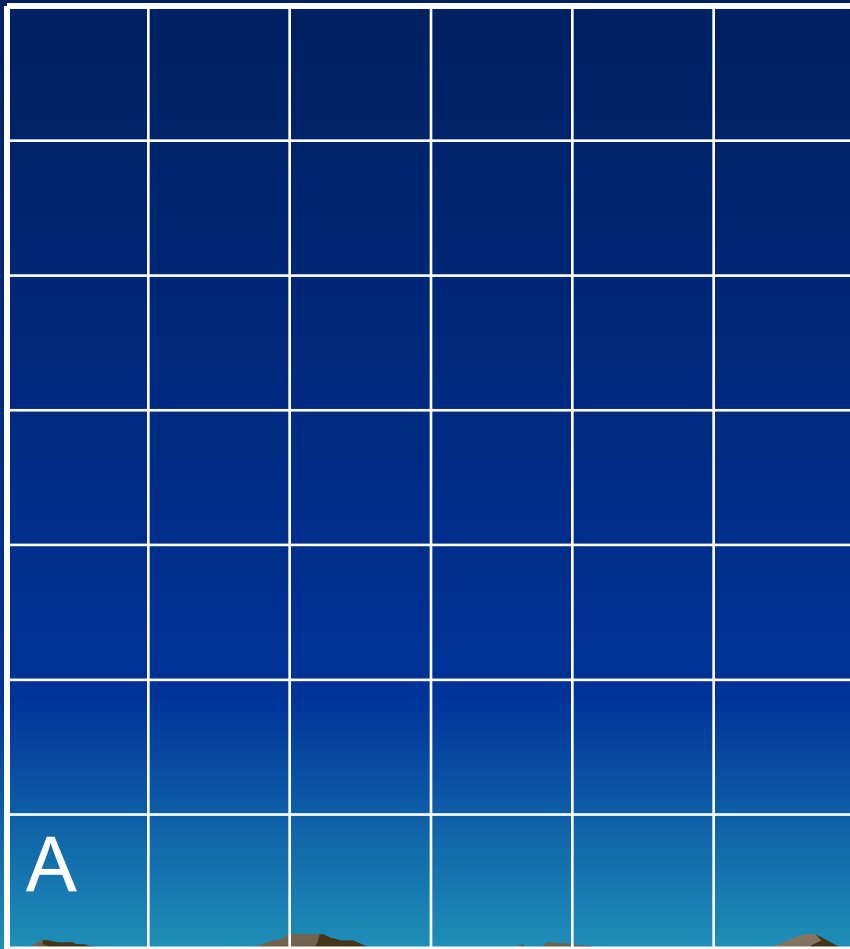
Chunk



A					

- `chunkNRows = 7`
- `chunkNCols = 6`
- A is a cell
 - It has a value like all others in the grid
- This chunk comprises 42 cells

Grid



- $nChunkRows = 7$
- $nChunkCols = 6$
- A is a chunk
 - It is made up of cells
- This Grid contains 42 chunks
- If these chunks each had 42 cells this would be 1764 cells

Why Chunk

- Each chunk can be stored optimally using any of a number of data structures
- Each chunk can be readily swapped and re-loaded as needs be
 - Memory handling



Different types of Chunk

- 64CellMap
- 2DArray
- JAI
- Map
- RAF



64CellMap (1/2)

- The most sophisticated data structure?
- Data stored in a fast, lightweight implementation of the java.util Collections API
 - gnu.trove
 - TdoublelongHashMap
 - TintlongHashMap
- The long gives the mapping of the value to the 64 cells in the chunk



64CellMap (2/2)

- For *chunks* that contain a single *cell* value there is a single mapping in the HashMap
- for *chunks* with 64 different *cell* values there are 64 mappings in the HashMap
- Iterating over (going through) the *keys* in the HashMap is necessary to get and set *cell* values, so generally this works faster for smaller numbers of mappings.



Map type chunks in general

- A mapping of *keys* (cell values) and *values* (cell identifiers) is a general way of storing *grid* data.
 - Efficient in terms of memory use where a default value can be set, and if there are only a small number of non-default mappings in the chunk (compared to the number of cells in the chunk)
 - Offer the means to generating some statistics about a chunk very efficiently
 - the diversity (number of different values)
 - mode



Factories and Iterators

- Each chunk and grid is associated with a factory and an iterator
- Factories keep things tidy, production can be done in one place and in a controlled way
- Iterators aim to offer the fastest and most efficient way of going through all the values in a grid or chunk
 - These can be ordered and unordered



Statistics (1/3)

- Attached to every grid is a statistics object
- Implemented by every statistics object and every chunk and grid is a statistics interface
- Abstract classes provide a generic way of returning statistics
- Specific chunks and grids can override these methods to provide faster implementations



Statistics (2/3)

- Two basic types attached to a grid
 - Updated
 - Statistics initialised and kept up to date as underlying data changes
 - Better the more often statistics are used
 - Not updated
 - Statistics not initialised or kept up to date as underlying data changes
 - Far faster if statistics are not used



Statistics (3/3)


- nonNoDataValueCountBigInteger
 - number of cells with non noDataValues
- sumBigDecimal
 - the sum of all non noDataValues
- minBigDecimal
 - the minimum of all non noDataValues
- minCountBigInteger
 - the number of min values as a BigInteger
- ... maxBigDecimal ... maxCountBigInteger



Memory Handling



```
/**
 * OutOfMemoryError Handling Wrapper for methodToProcess(args)
 * @param args Arguments needed for processing
 * @param handleOutOfMemoryError
 *   If true then OutOfMemoryErrors are handled in this method by
 *   calling swap(args) prior to recall of this method.
 *   If false then OutOfMemoryErrors are caught and thrown.
 */
public Object[] methodToProcess(
    Object[] args,
    boolean handleOutOfMemoryError ) {
    try {
        return methodToProcess( args );
    } catch ( java.lang.OutOfMemoryError e ) {
        if ( handleOutOfMemoryError ) {
            swap(args);
            return methodToProcess( args, handleOutOfMemoryError );
        } else {
            throw e;
        }
    }
}
```



Controlling Swap

- Swapping a chunk with values that are needed by the method could leave us in an infinite loop
- Swapping a chunk with values that are needed soon is not efficient if other chunks could have been swapped
- It is difficult to have a generic swap operations for all methods that is efficient
- When processing there can be multiple grids and it can be better to swap chunks in output grids or coincident chunks, or chunks in one grid then the next etc...
- The programmer knows best...



process

- Key Methods
 - addToGrid
 - aggregation
 - value replacement
 - mask
 - arithmetic operators
 - subtract
 - multiply
 - divide
 - rescaling
 - GWS
 - DEM



GWS

- Weighting
 - kernels
- Normalisation
- Multi scale generalisation
- 2 main types:
 - Univariate
 - First order
 - mean, sum
 - Second order
 - moments (proportions, variance, skewness)
 - Bivariate
 - difference
 - normalized difference
 - correlation



DEM extension

- Methods
 - Hollow or pit detection
 - Hollow filling
 - Flow accumulation
 - Distributive
 - Based on all downslope cells not just maximum
 - Geomorphometrics
 - E.g. Slope and aspect
 - Regional based and weighted like GWS



Processing Grids

- Simple case
 - A single input grid and a single numerical result
- Complex case
 - Multiple input and output grids
 - Grids of different
 - Sizes
 - Origins
 - Orientations



Multiple input and output Grids

- All Grids hold a reference to a collection of all the Grids
 - Used for swapping data



More about processing...

- Often involves generalising all cell values that lie within specific distances
- Often uses a distance weighting scheme
- Often involves producing outputs at the same resolution as inputs
- Often takes hours...
- Is the main reason for developing Grids...



Future Directions



- Handling different types of cell value
 - So far int and double type cell values only
 - Next want boolean and BigDecimal
- Use a virtual file store
 - To distribute swap across multiple networked machines
 - SRB?
- Take advantage of Java 1.5
- Organise for parallel processing using MPJ
- Enhance suite of geographical analysis methods



- eScience Collaboration with China
 - Develop as a Grid Service
- Develop unit tests
 - Key to opening up development?
- Improve documentation



Summary



Grids 1.0 beta is designed to handle

- Multiple input and multiple output Grids
- Grids with millions of rows and millions of columns
- Numerical data
- Grids containing chunks of the same dimensions



State Summary

- Not really taking advantage of Java 1.5
 - Currently based on JDK 1.4.2
 - plus a few handy extras
- Not developing fast
- Not abandoned
- Not really openly developed
 - Users encouraged to feedback and the problems get fixed by me...



Thank You For Your Attention!

Questions Advice Comments

<http://www.geog.leeds.ac.uk/people/a.turner/>



Acknowledgements

- The European Commission has supported this work under the following contracts:
 - IST-1999-10536 (*SPIN!-project*)
 - EVK2-CT-2000-00085 (*MedAction*)
 - EVK2-CT-2001-00109 (*DesertLinks*)
 - EVK1-CT-2002-00112 (*tempQsim*)
- The **ESRC** has supported this work under:
 - RES-149-25-0034 (*MoSeS*)
- Thank you James MacGill and Ian Turton for making available a version of the **GeoTools** Raster class Java source code which initially got me going with this package in January 2000.
- Thank you **University of Leeds** especially the **School of Geography** and **CCG** for your support and encouragement over the years.

