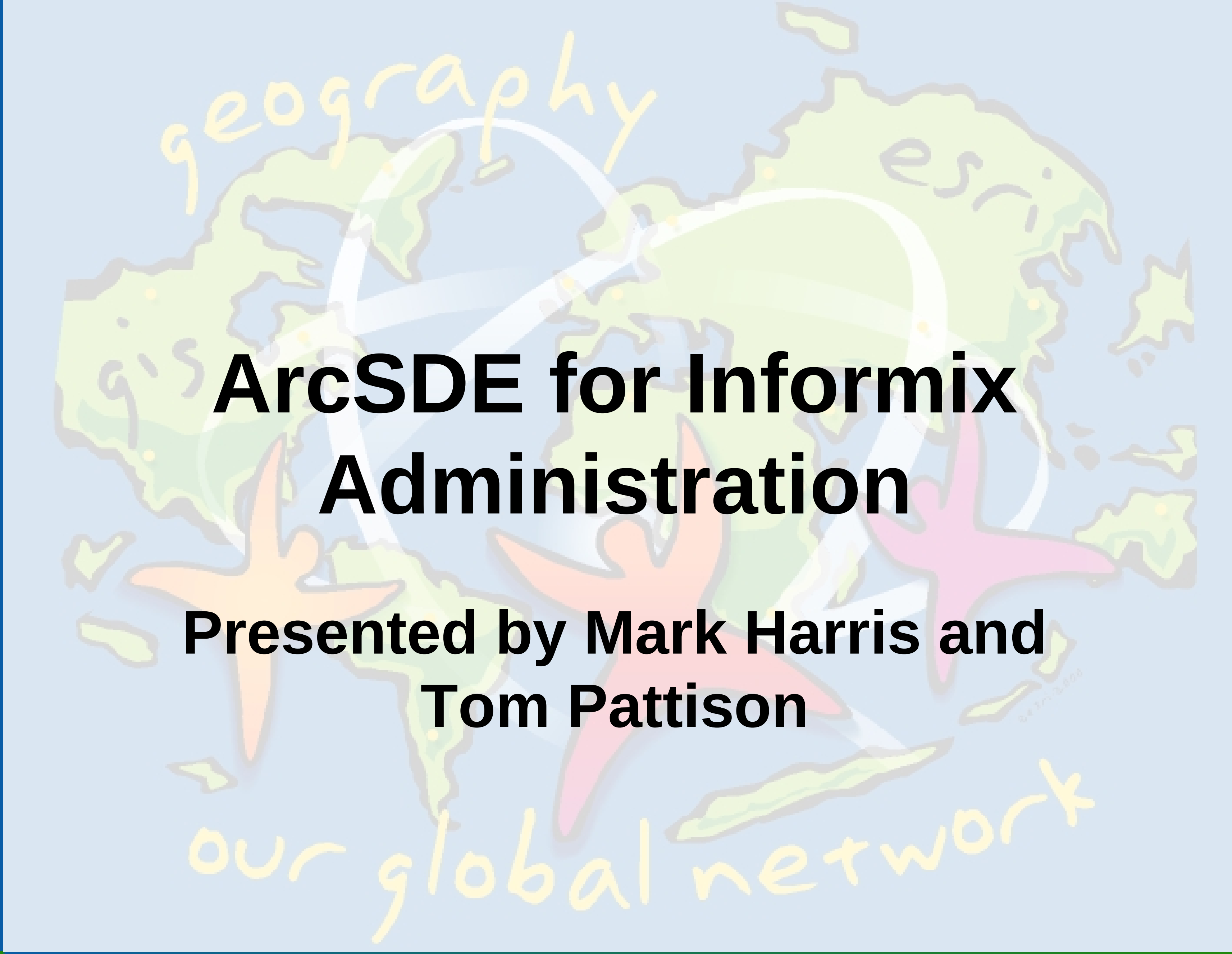


ArcSDE for Informix Administration

Presented by Mark Harris and Tom Pattison

Outline

- **Basic Tuning**
  - Tuning memory
  - Tuning CPU
  - Reducing Disk I/O contention
- Dbtune table
- Statistics
- Informix Spatial DataBlade
- Configure ArcSDE for loading data

Basic Informix Tuning

- Allocate and share CPU and Memory resources among various processes running on the server
- Adjust initialization parameters stored in the Onconfig file.
  - Must restart the instance to take affect
- Distribute "hot" files across file system to reduce disk I/O contention

Outline

- Basic Tuning
  - **Tuning memory**
  - Tuning CPU
  - Reducing Disk I/O contention
- Dbtune table
- Statistics
- Informix Spatial DataBlade
- Configure ArcSDE for loading data

Tuning Memory

- Memory is allocated to:
  - the Informix servers shared buffers
  - the Informix server processes
  - the ArcSDE giomgr process
  - the ArcSDE gsrvr processes
  - the O/S
  - other processes

Regular buffer

- Configured by the BUFFERS parameter
- Holds MRU data pages
- Set buffers to 25 percent of physical RAM
- Buffers are set in pages
- Example: to calculate the buffers for 256MB of RAM and a 2 kilobyte page

*(256 * 1024) * 0.25 / 2*

Regular buffers

- Examine the %cached reads of the onstat -p command after server is warm
- Taking current memory use into account, if it is below 90% increase BUFFERS

LOGSIZE

- The LOGSIZE parameter controls the size of the logical log files
- Increase the LOGSIZE parameter to 1500

LOGBUFF

- Size of the buffers that buffer writes to the logical logs
- There are three buffers
- Defaults to 32 kilobytes
- Set to even increment of Informix pagesize

LOGBUFF

- After the server is warm, use onstat -l and examine Logical Logging section
  - if pages/io is less than 75% of bufsize reduce LOGBUFF
  - if pages/io is greater than 95% of bufsize increase LOGBUFF

PHYSBUFF

- The size of the buffers that buffer writes to the physical log
- There are two physical log buffers
- Set to an even increment of the Informix pagesize

PHYSBUFF

- After the server is warm, use onstat -l and examine Physical Logging section
  - if pages/io is less than 75% of bufsize reduce PHYSBUFF
  - if pages/io is greater than 95% of bufsize increase PHYSBUFF

LOGSMAX

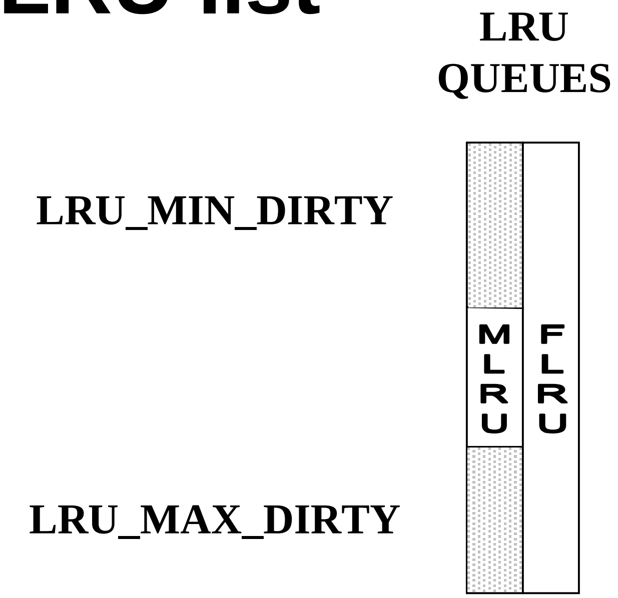
- The LOGSMAX parameter controls the maximum number of logical logs that can be created
- Increase this parameter to create the new log files in the separate log file dbspaces

CLEANERS

- The CLEANERS parameter controls the number of page cleaners
- Page cleaners write dirty blocks in the regular buffer to disk
- Set the page cleaners to the number of frequently accessed disks
- Set to at least 6

Least recently used queues

- LRU queue pair maintain a FLRU and a MLRU list



LRU queues

- Avoid foreground writes by having enough LRU queues
- The number of LRU queues is controlled by the LRUS parameter
- Set LRUS to 4 times the number of CPUs

LRUs and page cleaners

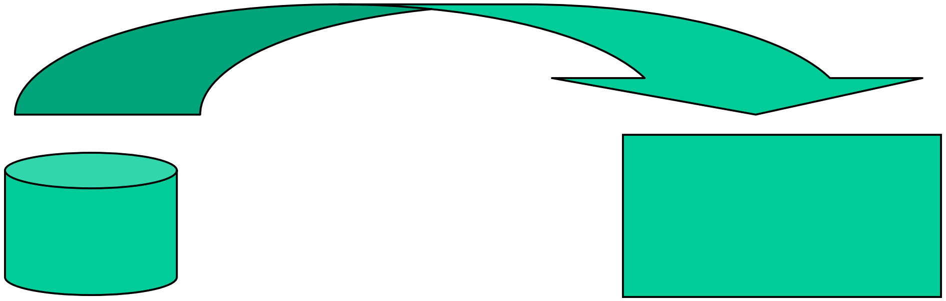
- Use onstat -F to monitor the number of foreground writes
- Increase LRUS and CLEANERS to reduce foreground writes

Shorter LRU queues

- Page cleaners wake up and sleep according to LRU_MAX_DIRTY and LRU_MIN_DIRTY
- Lower them to shorten queues and reduce foreground writes
  - only if increasing LRUS and CLEANERS did not help

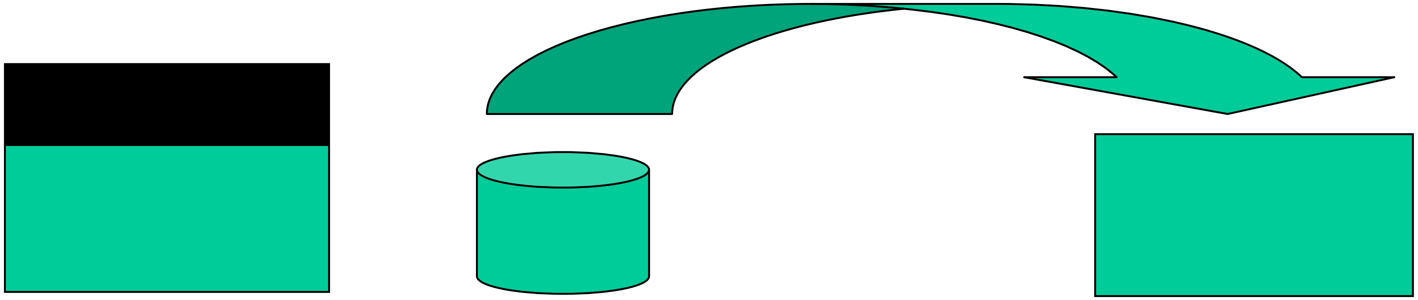
RA_PAGES

- The RA_PAGES parameter controls the number of pages read ahead
- Increase RA_PAGES to 125



RA_THRESHOLD

- RA_THRESHOLD - number of unprocessed pages that trigger a read ahead
- Increase the RA_THRESHOLD to 85



Outline

- Basic Tuning
  - Tuning memory
  - **Tuning CPU**
  - Reducing Disk I/O contention
- Dbtune table
- Statistics
- Informix Spatial DataBlade
- Configure ArcSDE for loading data

Tuning CPU

- For single CPU servers
  - Set SINGLE_CPU_VP to 1; bypass superfluous mutex calls
  - Set MULTIPROCESSOR to 0; optimize locking for single processor
  - These are the defaults

Tuning CPU

- For multiple CPU servers
  - Set SINGLE_CPU_VP to 0
  - Set MULTIPROCESSOR to 1
  - Take advantage of the machines parallel processing capabilities
  - Note: may want to treat a dual-processor as a single processor

Virtual Processors

- Kind of like O/S processes
- Actually threads of the Informix oninit process
- Manage background tasks and service client application

Virtual Processors

- Virtual processor classes that are tunable include
  - CPU
  - AIO
  - Network

Tuning VPs

- AIO and CPU class can be tuned by setting:
  - NUMCPUVPS
  - AFF_SPROC
  - AFF_NPROCS
  - NOAGE
  - NUMAIOVPS

Tuning VPs

- This presentation describes the use of the VPCLASS parameter
- You cannot mix the VPCLASS with the other VP parameters
  - An error results when the server is initialized if both types of parameters are present in the onconfig

User defined VPCLASS

- Informix allows you to define your own VP class
- You should not, unless an Informix DataBlade requires you to do so
- The Informix Spatial DataBlade uses the predefined CPU VP class

CPU and AIO VP class

- Informix always has at least one AIO VP class defined to handle ancillary tasks such as messaging

```
VPCLASS aio, num=1
```

CPU and non-logged I/O on UNIX Systems

- UNIX systems must use the AIO VP for nonlogged I/O
  - if kernel-asynchronous I/O (KAIO) is not implemented
  - if the I/O performed is to a cooked file
- Otherwise they use CPU VP
  - Advantage: Less context switching between AIO and CPU VPs

CPU VP and logged I/O on UNIX systems

- UNIX systems must use PIO and LIO to perform physical and logical logging
  - if kernel-asynchronous I/O (KAIO) is not implemented
  - if the log files are stored on cooked file
- Disadvantage:
  - context switching between CPU VP and PIO and LIO VP

Raw devices and KAIO on UNIX

- Advantages:
  - – reduced context switching between CPU and other VPs
  - – eliminates I/O to the O/S buffer
- Disadvantages
  - – Difficult to setup and maintain

CPU and AIO VP class on Windows NT

- Windows NT systems always use CPU virtual processor to perform both logged and nonlogged I/O
- Set at least one AIO VP and several CPU VPs

CPU VPCLASS

- Do not set the CPU VPs higher than number of processors on machine
- Start the CPU VPs at 2
- Example:

```
VPCLASS cpu, num=2, max=8
```

AIO VPCLASS on UNIX

- If KAIO is implemented and all non-logged I/O is to raw device, configure at least one AIO VP

```
VPCLASS aio, num = 1
```

AIO VPCLASS on UNIX

- If KAIO is implemented and some cooked devices are used, allocate two AIO VPs per active dbspace composed on a cooked device

```
VPCLASS aio, num = 8
```

AIO VPCLASS on UNIX

- If KAIO is not implemented, allocate two AIO virtual processors for each frequently accessed disk

```
VPCLASS aio, num = 16
```

Network Virtual Processors

- The NETTYPE parameter defines the number of poll threads allocated to each connection type
- Poll threads can be run by either network VPs or CPU VPs

Network virtual processors on UNIX systems

- Poll threads tend to run more efficiently on CPU VPs; good for single CPU server
- If CPU VPs become congested you need to offload the poll threads to Network VPs

Network Virtual Processors on UNIX systems

- This is a typical NETTYPE configuration.

```
NETTYPE    ipcshm,1,20,CPU
NETTYPE    tlitcp,2,100,NET
```

Network Virtual Processors on Windows NT

- The NETTYPE vp-class field is always set to NET on Windows NT even though the poll threads are handled inline

```
NETTYPE onsoctcp, 1,500,NET
```

VP Priority aging

- You should disable priority aging
- Enabled by default
- VPs run continuously and run with a lower priority the longer the server is up, unless priority aging is disabled

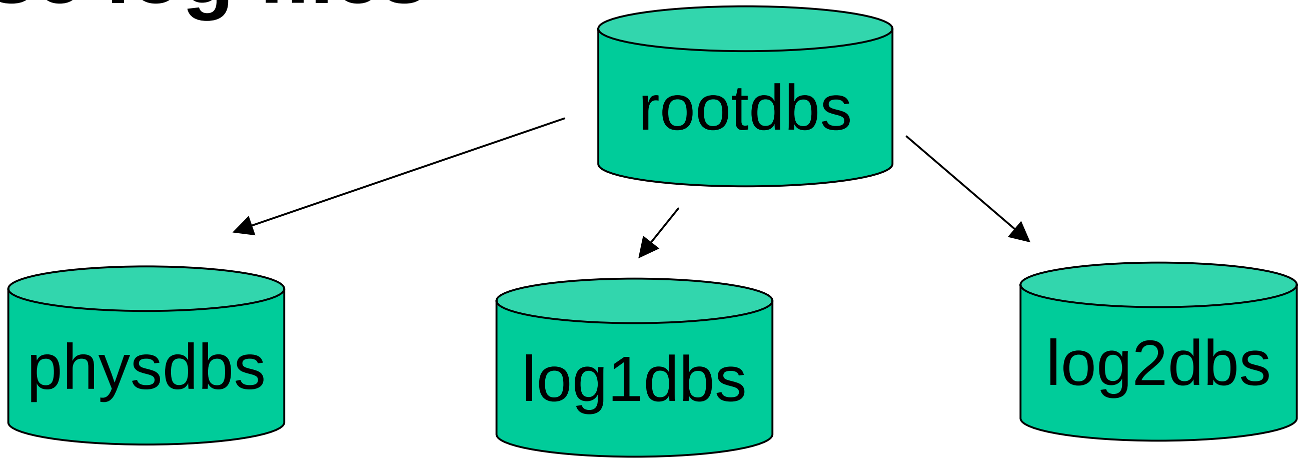
```
VPCLASS CPU, num=2, max=8, noage
```

Outline

- Basic Tuning
  - Tuning memory
  - Tuning CPU
  - **Reducing Disk I/O contention**
- Dbtune table
- Statistics
- Informix Spatial DataBlade
- Configure ArcSDE for loading data

Log file dbspaces

- Physical and logical log are created in the root dbspace
- Create separate dbspaces for each of these log files



Log file dbspaces

- Position physical and logical logs dbspaces on separate disk drives
- May position one or the other with the root dbspace if necessary

Physical log file

- Creating the physical log dbspace

```
onspaces -c -d physdbs -p E:\inf_data\physdbs.000 -o 0 -s 10000
```

- Set the PHYSDBS and PHYSFILE configuration parameters

```
PHYSDBS         physdbs
PHYSFILE        9000
```

Move the logical log out of the rootdbs

- To move the logical log to its own dbspaces
  - Create logical log file dbspaces
  - Add logical log to new dbspaces
  - Move current logical log to a new log
  - Remove old logical logs

Create new logical logs

- Create the logical log dbspace

```
onspaces -c -d log1dbs -p D:\inf_data\log1dbs.000 -o 0 -s 10000
```

- Create the new logs

```
onparams -a -d log1dbs
```

- Enable the new logs

```
onmode -s
ontape -s
```

Remove the old logs (1)

- Get the current log

```
onstat -l
address  number  flags    uniqid   begin    size   used   %used
a049784  1       U---C-L  1367     200035   750    750    57.00
a0497a0  2       U-B----  1368     200323   750    750     0.00
a0497bc  3       U-B----  1369     200611   750    750     0.00
a0497d8  4       U-B----  1370     2008ff   750    750     0.00
a0497f4  5       U-B----  1371     200bed   750    750     0.00
a049810  6       U-B----  1372     200edb   750    750     0.00
a04982c  7       U-B----  1373     2011c9   750    750     0.00
a049848  8       U-B----  1374     2014b7   750    750     0.00
a049864  9       U-B----  1375     2017a5   750    750     0.00
a049880  10      U-B----  1376     201a93   750    750     0.00
```

← In rootdbs

Remove the old logs (2)

- Switch the current log

```
onmode -l;onmode -l;onmode -l;onmode -l;onmode -l;onmode -l
onstat -l
```

```
address  number  flags    uniqid  begin    size  used   %used
a049784  1       U-B----  1367    200035   750   750    57.00
a0497a0  2       U-B----  1368    200323   750   750     0.00
a0497bc  3       U-B----  1369    200611   750   750     0.00
a0497d8  4       U-B----  1370    2008ff   750   750     0.00
a0497f4  5       U-B----  1371    200bed   750   750     0.00
a049810  6       U-B----  1372    200edb   750   750     0.00
a04982c  7       U---C-L  1373    2011c9   750   750     0.00
a049848  8       U-B----  1374    2014b7   750   750     0.00
a049864  9       U-B----  1375    2017a5   750   750     0.00
a049880  10      U-B----  1376    201a93   750   750     0.00
```

In rootdbs

Remove the old log (3)

- Delete the old logs

```
ontape -s
onparams -d -l 1 -y
onparams -d -l 2 -y
onparams -d -l 3 -y
onparams -d -l 4 -y
onparams -d -l 5 -y
onparams -d -l 6 -y
```

Remove the old logs (4)

- The remaining logs

```
onstat -l
address  number  flags    uniqid   begin    size  used   %used
a049784  7       U---C-L  1367     200035   750   750    0.00
a0497a0  8       U-B----  1368     200323   750   750    0.00
a0497bc  9       U-B----  1369     200611   750   750    0.00    ← In log2dbs
a0497d8  10      U-B----  1370     2008ff   750   750    0.00
a0497f4  11      U-B----  1371     200bed   750   750    0.00
a049810  12      U-B----  1372     200edb   750   750    0.00
a04982c  13      U-B----  1373     2011c9   750   750    0.00
a049848  14      U-B----  1374     2014b7   750   750    0.00    ← In log2dbs
a049864  15      U-B----  1375     2017a5   750   750    0.00
a049880  16      U-B----  1376     201a93   750   750    0.00
```

Temporary space

- Temporary space is used for:
  - – creating indexes
  - – sorting
  - – performing joins

Temporary dbspaces

- By default temporary storage is written to /tmp or C:\tmp
- Create the temporary dbspaces

```
onspaces -c -t -d temp1dbs -p F:\inf_data\temp1dbs.000 -o 0 -s 20000
```

- Set DBSPACETEMP

```
DBSPACETEMP temp1dbs, temp2dbs
```

Default smart large object

- The Informix Spatial DataBlade stores compressed geometry that is larger than 930 bytes in “smart blobs”
- Create the sbspace

```
onspaces -c -s sblobdbs -g 1 -p F:\inf_data\sblobdbs.000 -o 0 -s 100000
```

- Set the onconfig parameter

```
SBSPACENAME          sblobdbs
```

Smart large object

- Separate the smart large object sbspace from the business dbspace containing the spatial columns

Tables and indexes

- Separate dbspaces for tables and indexes
- Position the dbspaces on different disk drives
- Provides simultaneous disk I/O to table and indexes

High use tables

- Position high use tables in the middle partitions of the disk drives
  - reduces disk head travel
- Separate large high use tables from one another by placing their dbspaces on separate disk drives

Group small tables

- Group smaller tables together by usage
- Allows you to mix “hot” dbspaces with “not-so-hot” dbspaces
- Remember to keep tables separate from indexes

Dbspaces for large tables

- Each large index or table (greater than 500MB) should be stored in its own dbspace
- May need multiple dbspaces for one object to distribute it around file system

Few extents as possible

- Keep the number of extents/object as low as possible
- Lot of extents increase the possibility of interleaving and disk head travel
- More extents mean more overhead

Outline

- Basic Tuning
  - Tuning memory
  - Tuning CPU
  - Reducing Disk I/O contention
- **Dbtune table**
- Statistics
- Informix Spatial DataBlade
- Configure ArcSDE for loading data

DBTUNE Configuration Parameters

- Configuration parameters communicate object storage information to INFORMIX
- Configuration parameters are grouped into keywords

The DBTUNE table

- At ArcSDE 8.1 the dbtune.sde file becomes the dbtune table
- At ArcSDE 3.x each dbtune parameter described a Informix parameter
- At ArcSDE 8.1 each dbtune parameter describes an Informix object

The ArcSDE 3.X keyword

```
##ROADS
    A_TBLSP              roadsdbs
    A_IX_TBLSP           road_ix_dbs
    A_SBLOB_DBS          roadsblob
    A_INIT               40
    A_NEXT               40
    A_IX_FILL            90
    A_LOCK_ROW           1
END
```

The ArcSDE 8.1 keyword

```
##ROADS
UI_TEXT        "General roadway storage"
COMMENT        "Do not change these settings -- BOB!!!!"
B_STORAGE      "in roadsdbs extent size 40 next size 40
                 lock mode row put feature in (roadsblob)"
B_RTREE        "in road_rt_dbs extent size 40 next size 40
                 fillfactor 90"
B_INDEX_1      "in road_ix_dbs extent size 40 next size 40
                 fillfactor 90"
END
```

Sdedbtune admin tool

- Imports a dbtune file into a DBTNE table
- Exports the DBTUNE table to a dbtune file
- Allows you to update the DBTUNE table by editing the dbtune file

Outline

- Basic Tuning
  - Tuning memory
  - Tuning CPU
  - Reducing Disk I/O contention
- Dbtune table
- **Statistics**
- Informix Spatial DataBlade
- Configure ArcSDE for loading data

Statistics

- Informix Dynamic Server uses Cost Based Optimization
  - You need to keep the data objects statistics up-to-date
- Use UPDATE STATISTICS to keep spatial data statistics up-to-date

Statistics

- Use UPDATE STATISTICS whenever more than 10-20% of the data has been modified
- Switching from Load Only I/O mode to Normal I/O mode automatically executes UPDATE STATISTICS

Statistics

- Use UPDATE STATISTICS on tables containing a spatial column
- Example

```
update statistics for table streets;
```

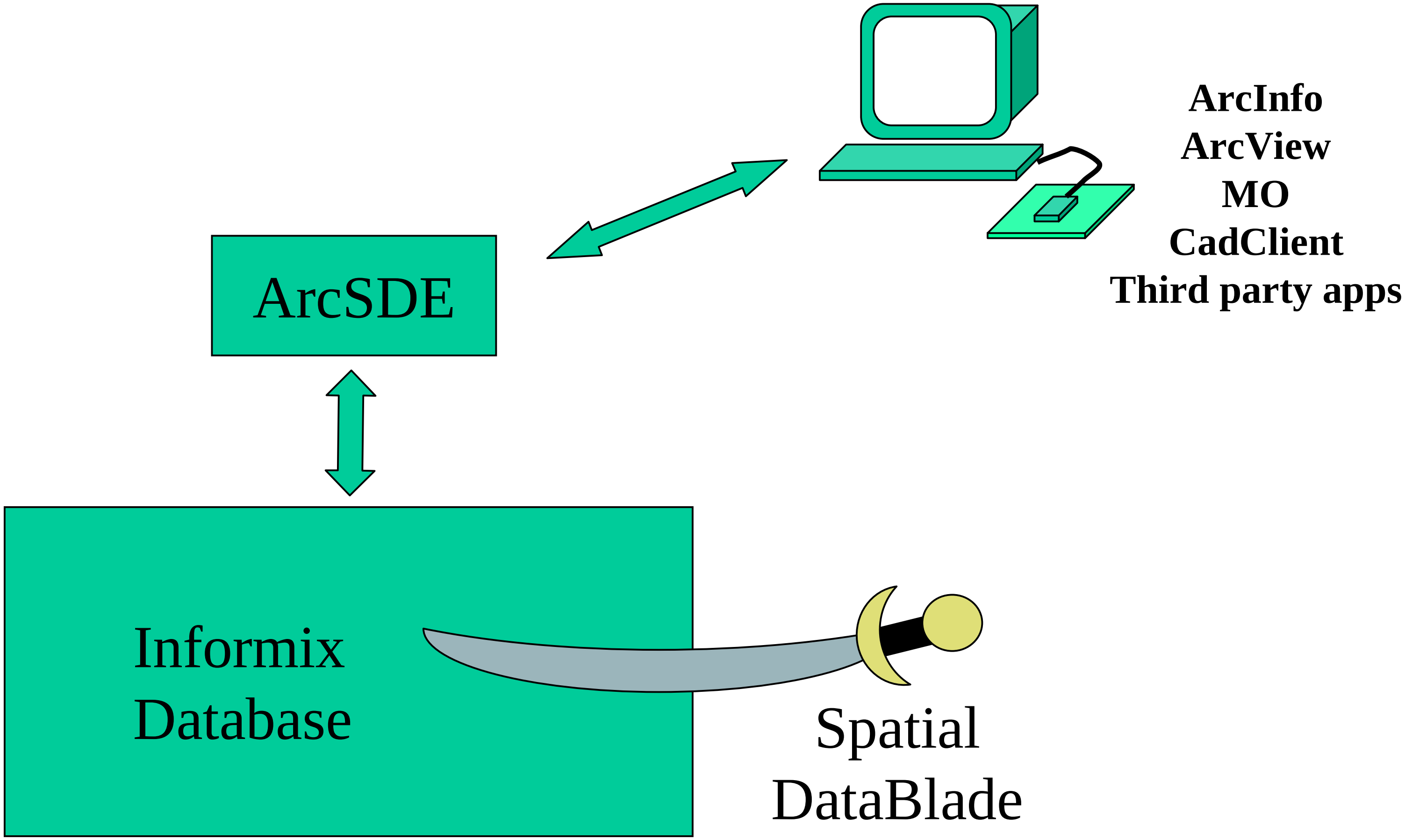
Outline

- Basic Tuning
  - Tuning memory
  - Tuning CPU
  - Reducing Disk I/O contention
- Dbtune table
- Statistics
- **Informix Spatial DataBlade**
- Configure ArcSDE for loading data

Informix Spatial DataBlade

- Joint effort of the Informix and ESRI partnership
- Adds OGC compliant spatial types and functions to an Informix database
- *ArcSDE for Informix* calls spatial functions and uses spatial types

Informix Spatial DataBlade



Informix Spatial Datablade

- The spatial datablade directory Spatial.8.1 is installed under Informix extend directory
- Register the Informix Spatial Datablade to all databases that store spatial data

Spatial Data Types

- Spatial data types include:
  - ST_Geometry
  - ST_Point
  - ST_LineString
  - ST_Polygon
  - ST_Multipoint
  - ST_MultiLineString
  - ST_MultiPolygon

```
Create table bldftprints
(building_ids,
 name varchar(32),
 feature ST_LineString);
```

Spatial functions

- Comparison predicates
  - ST_Equals
  - ST_OrderingsEquals
  - ST_Touches
  - ST_Overhaps
  - ST_Crosses
  - ST_Within
  - ST_Contains
  - ST_Disjoint
  - ST_Intersects
  - ST_Relate
  - SE_EnvelopesIntersect

```
Select building_id
from streets sts,bldftprints bld
where ST_Intersects(sts.feature,bld.feature) = ‘t’;
```

Spatial functions

- Property predicates
  - ST_IsRing
  - ST_IsClosed
  - ST_IsSimple
  - ST_IsEmpty
  - ST_IsValid
  - SE_Is3D
  - SE_IsMeasure

```
Select count(*) from bldftprints where
ST_IsRing(feature) = ‘f’;
```

Spatial functions

- Property functions
  - ST_Area
  - ST_Boundary
  - ST_CoordDim
  - ST_Distance
  - ST_EndPoint
  - ST_Envelope
  - ST_Dimension
  - ST_ExteriorRing
  - ST_GeometryType
  - ST_Length
  - ST_NumGeometries
  - ST_NumInteriorRing
  - ST_NumPoints
  - ST_Perimeter
  - ST_SRID
  - SE_M
  - ST_X
  - ST_Y
  - SE_Z

Spatial functions

- Text converters
  - ST_GeomFromText
  - ST_PointFromText
  - ST_LineFromText
  - ST_PolyFromText
  - ST_MPointFromText
  - ST_MLineFromText
  - ST_MPolyFromText
  - ST_WKTToSQL
  - ST_AsText

```
CREATE TABLE wells (g1 ST_Geometry);

INSERT INTO wells
VALUES (GeometryFromText (‘point (10.02 20.01)’,1));
```

Spatial functions

- Well Known Binary Converters
  - ST_GeomFromWKB
  - ST_PointFromWKB
  - ST_LineFromWKB
  - ST_PolyFromWKB
  - ST_MPointFromWKB
  - ST_MLineFromWKB
  - ST_MPolyFromWKB
  - ST_WKBToSQL
  - ST_AsBinary

Spatial functions

- (binary) Shape Converters
  - SE_GeomFromShape
  - SE_PointFromShape
  - SE_LineFromShape
  - SE_PolyFromShape
  - SE_MPointFromShape
  - SE_MLineFromShape
  - SE_MPolyFromShape
  - SE_ShapeToSQL
  - SE_AsShape

Spatial functions

- Functions that generate geometry
  - ST_Union
  - ST_Intersection
  - ST_Difference
  - ST_SymmetricDiff
  - ST_Buffer
  - ST_Centroid
  - ST_ConvexHull
  - ST_GeometryN
  - ST_InteriorRingN
  - ST_Point
  - ST_PointN
  - ST_PointOnSurface
  - ST_Polygon
  - ST_Transform

Spatial functions

- Network Functions
  - SE_LocateAlong
  - SE_LocateBetween

```
SELECT LocateBetween(roads,surface.from,surface.to)
FROM highways, surface
```

The RTREE index

- The Informix Spatial DataBlade indexes the spatial column with an RTREE index.
- No fuss, no muss

```
CREATE INDEX <name> on <table>
   (<column_name> St_Geometry_Ops) using Rtree;
```

Outline

- Basic Tuning
  - Tuning memory
  - Tuning CPU
  - Reducing Disk I/O contention
- Dbtune table
- Statistics
- Informix Spatial DataBlade
- **Configure ArcSDE for loading data**

Configure ArcSDE for loading

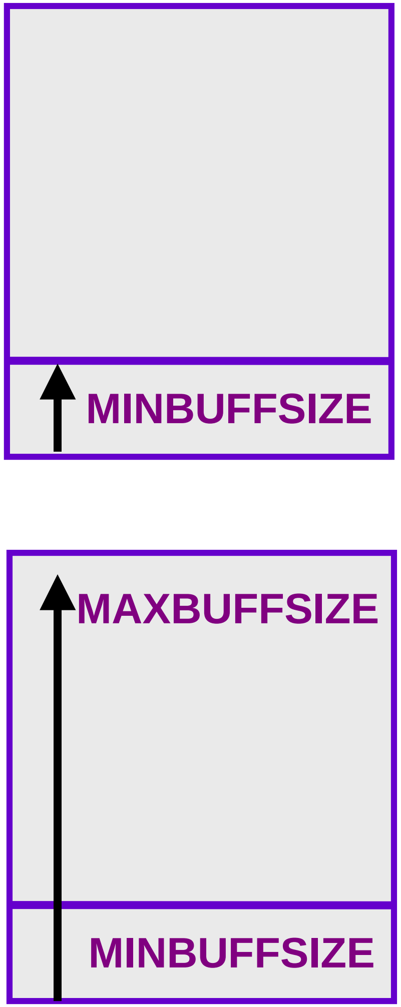
- Use the `giomgr.defs` file to alter ArcSDE configuration for data loading
- ArcSDE uses *transmission buffers* to support asynchronous I/O and streamline network usage

ArcSDE Transmission Buffers

- Buffers are allocated on the server side and the client side
  - The size of the transmission buffer is set with `MAXBUFFSIZE` and `MINBUFFSIZE`
- Larger buffers allow higher throughput increasing performance during bulk loading

ArcSDE Transmission Buffers

- If the server is waiting, the buffer will gather up to MINBUFFSIZE of data to send to the server
- If the server is busy the client will gather up to MAXBUFFSIZE of data before sending the buffer to the client
- Reduces I/O by batching a minimum of amount of data



ArcSDE Transmission Buffers

- High transmission buffers increase performance during bulk loading
- Do not use high buffers for normal query operations
  - ArcSDE will assign the `MAXBUFSIZE` amount of memory to each stream and may exhaust available memory

Autocommit

- The giomgr.defs default commit interval is 1000 records
- Increase autocommit interval to 5000 for loading

Where to get more help

- Informix classes
- ESRI classes
- Informix technical support
- ESRI technical support

Questions?

- Please don’t forget to fill out the survey before you leave

ESRI