IRI CoSort

Sort, Transform & Report

Best Practices

A Supplement to the CoSort Version 10 Product Installation and User Guides

Confidential

Following are best practice recommendations for CoSort Version 10, representing the collective experience of IRI support personnel and end users. This supplement is not intended to replace the CoSort package documentation (User and Installation Guides), or the in-dialog and Help Menu contents available in IRI Workbench.

Please email **support@iri.com** *with any questions, and to provide your feedback for improvements to this document.*

CoSort Installation

1. CoSort is shipped as an InstallShield executable on Windows and as a g'zipped tar file on Linux and Unix. Save the installer in case you need it in the future.
2. Follow any special instructions provided by email from your IRI agent. Review the readme file (release notes) and the .pdf installation guide.
3. CoSort also ships with the free 'IRI Workbench' front-end for SortCL job design and management, built on Eclipse. Follow all instructions in the installation guide (or email you received) to unpack and update Workbench after licensing CoSort.
4. Read the Welcome content (also available from the Workbench Help menu).
5. Access to CoSort licensing and tuning parameters requires the existence of the $COSORT_HOME (or %) environment variable, which must be set to your CoSort installation directory. Ensure that each CoSort user has this access.

CoSort Licensing

1. CoSort licenses are tied to the hostname. Therefore, for each new physical or virtual machine where CoSort is installed, a new set of license keys is required.
2. For each CoSort installation, select the *First Time Set*up option, and when asked "*Have you received your keys from IRI?*" answer "*no*". This will prompt you to complete a form that will generate the *RegForm.txt* file. Email this file to licenses@iri.com to secure your license keys. CoSort will NOT run with license keys generated from a RegForm.txt that was created on a different platform.
3. You must have writing, editing, and reading permission on the cosort/etc/cosort.lic file. Otherwise the setup program cannot insert your keys.
4. In order to test for a valid license, make sure that the COSORT_HOME environment variable is pointing to the correct CoSort install directory. On Unix, execute *$COSORT_HOME/bin/sortcl /v*. On Windows, enter the *<cosort_install_dir>\bin* on the command line and execute *sortcl /v*.
5. When there are multiple versions of CoSort on the same machine, make sure that your $COSORT_HOME environment value points to the correct version in order to avoid licensing errors.

CoSort Tuning Parameters

1. See the Appendix chapter, Section D, of the CoSort User Guide and these FAQs.
2. Default resource settings are in the *$COSORT_HOME/etc/cosortrc* file on Unix/Linux, and the Windows Registry or the cosort.rc file on Windows. Use a text editor (vi, notepad) or the CoSort Workbench wizard or diaglogs to edit files.
3. Consider the use of job-specific cosortrc files as well; see the /MEMORY-WORK command and the COSORT_TUNER environment variable. IRI Workbench also allows you to modify, monitor, and create local .cosortrc (Unix) or cosort.rc (Windows) files to override the global settings.
4. CoSort accepts the settings under the assumption that only one CoSort job using these values is running. Therefore, you should reduce the number of threads and the values for memory and BLOCKSIZE in proportion to the number of CoSort jobs running concurrently. For example, consider lowering these values by 2/3 if you are executing three jobs at the same time.
5. THREAD_MAX should be set to the number of threads paid for when licensing, or requested during the trial of, CoSort on that host. The error “Maximum Number of Sort Sub-Processes Exceeded” indicates THREAD_MAX is set to a value higher than that allowed by your license.
6. Normally, the MEMORY_MAX value assigned during setup (AUTO) is optimal. This value is the amount of RAM used before temporary files are written to disk. Sorts that run in memory are faster than those that write temporary files. There are some cases where increasing this value allows sorts to run fully in memory.
7. On Unix/Linux, be sure the values reflected in each user's *ulimit –a* command are compatible with your cosortrc settings. If not, memory errors can occur.
8. BLOCKSIZE must be a multiple of 4KB (4096). The minimum is 68KB.
9. When WORK_AREAs are separate from where the input and output will reside, a smaller BLOCKSIZE is better.
10. When input, output, and temporary files are on one disk, try a larger blocksize (between 256KB and 2MB).
11. If you get an error for insufficient merge memory, decrease BLOCKSIZE.
12. For WORK_AREAs, assign a different hard disk drive (HDD) for each thread, up to the value of THREAD_MAX.
13. WORK_AREA assignments are cumulative. If there are WORK_AREA assignments in the main cosortrc file, and more in another cosortrc file that is being accessed, then all WORK_AREAs from both cosortrc files will be used.
14. If possible, do not specify a HDD for WORK_AREA if that HDD will contain the input or output files, as that will cause runtime I/O contention.
15. Allocate 3-4x the largest input size for input, temp and output file space.
16. Temporary space requires at least 1x the size of the input file.
17. Experiment with different THREAD_MAX values with your trial license; ask IRI at the outset for a license key that will accommodate the total number of cores aboard. Once you find the optimal THREAD_MAX setting for each node, you should request it in the permanent license key.
18. For jobs with a small amount of input data, performance may improve with a lower THREAD_MAX value (try a setting of 2).

Improving CoSort Performance and Streamlining Processes

1. Use pipes rather than files between processes (if possible) to avoid intermediate file writes between job stages (since file I/O is often a performance bottleneck).
2. Use batch scripts to automate processing.
3. Schedule SortCL jobs from IRI Workbench, cron, Stonebranch UAC, Autosys, or another job control system. The Full360 MetaController checks for inter-job dependencies, and can be expanded to modify SortCL parameters directly.
4. Use environment variables whose values may be set at runtime. Remember that on Unix and Linux, environment variables must be exported before they will be recognized.
5. Up until a file size 1.5 times the value of MEMORY_MAX, CoSort tries to do an in-memory sort, which is faster. Thus, if your file size is close to that mark, you may want to increase your MEMORY_MAX setting.
6. Learn to leverage CoSort to accelerate existing tools and systems. See the http://www.iri.com/solutions area and the ETL/DB Acceleration section in particular, for application-specific performance advice.

SortCL Application Performance Tips

1. Use SortCL to combine multiple data transformation and formatting operations in the same job script and I/O pass, such as an unsorted join with aggregation breaks on the join key (to be sorted).
2. Specify multiple custom-formatted targets at the same time, along with field-level data masking functions, like format-preserving encryption on credit-card fields, so that the output data be outsourced or used for privacy-law-compliant testing.
3. Specify report formats on output to eliminate the need for additional passes.
4. If input or output records are fixed-length, specify a /LENGTH for the file so CoSort does not have to waste time seeking or writing line terminators.
5. Where possible, specify selection criteria with /INCLUDE or /OMIT statements in the input phase (rather than in the output phase) to reduce data for the action (e.g. sort) phase. By not processing unnecessary records, you reduce I/O and increase the efficiency of the action phase.
6. Specify ASCII collation instead of NUMERIC where it will not affect the results because ASCII sequencing is faster.
7. If you have multiple or compound include or omit statements, or multi-condition field definitions, structure the condition so that the statement that is most likely to evaluate as true is first. This saves the time spent evaluating the other parts of the compound condition.
8. During a two-file join, define the larger file as the left file.

CoSort Logging Options

1. The cosort.log file stores information about previously run jobs. The location of the cosort.log file is set in the cosortc "LOG" entry.
2. The AUDIT resource setting allows you to record more detailed information about previously executed SortCL jobs. The audit trail is stored in an XML file that you can query with an XML query tool or SortCL. A SortCL data definition file (.ddf) for the AUDIT log format is provided in the examples sub-directory.
3. Statistics can be obtained for each SortCL run with the /STAT command.

CoSort Error Checking

1. Refer to the User Guide, Appendix E, for error numbers and meanings.
2. Upon receiving an error, review the .cserrlog file for more details about where the error occurred. The .cserrlog file is located in the directory pointed to by the cosortrc LOG variable. Make sure that the user running CoSort has write permission for this directory so in case of an error, the .cserrlog can be written.
3. Syntax errors generated during SortCL execution indicate the line to fix.

Writing CoSort (SortCL) Job Scripts

Translating Third-Party Metadata

1. You may have existing data definitions in COBOL FD, MVS and VSE JCL sort, CSV file, ELF, and/or Oracle SQL*Loader control file (.ctl) formats. Those can be translated into SortCL syntax using the **2ddf* translators in the *$COSORT_HOME/bin* directory or via the Import/Convert wizard in IRI Workbench. See the SORTCL Tools chapter in the manual.
2. SortCL DDFs can also be created automatically via erwin Mapping Manager and Meta Integration Model Bridge (MIMB) software, plus IRI Fast Extract (FACT), and the TeraStream ETL tool (from DataStreams Korea).
3. Re-use SortCL metadata (.ddf) and jobs specification (.scl) files by creating a commonly used metadata and specification file repository. IRI Workbench facilitates the creation, editing, and re-use of DDF repositories through project folders, form editors, and asset management (source code) plug-ins like Git,
4. Though specification files can be nested to several levels, no more than two layers are recommended.
5. IRI Workbench converts between SQL DDL and SortCL DDF metadata, and can produce CSV and DB load utility configuration files from DDF /FIELD sections.

Using the SortCL Job Script Editor in IRI Workbench

1. See the advice from Eclipse here, and IRI tips and tricks here.
2. Learn keyboard shortcuts for the syntax-aware SCL editor, such as:
 F3 (open reference), Ctrl-F (find), Ctrl-L (go to line), Ctrl-Shft-Space (context-sensitive hints), Ctrl-Shift-F (reformat), Ctrl-\ (comment or uncomment)
3. Learn more about job design generally in this self-learning section.

General Scripting Recommendations

1. Make sure that job scripts are well commented, and that indents occur at the correct places to improve readability.
2. It is important to understand the file type and record layout structures of data before writing the DDFs and job scripts. CoSort supports a wide range of file formats and data types. Refer to the User Guide for detailed information on these, so that they may be correctly identified in the job and metadata scripts.
3. There are special indexed file types like VSAM and MF-ISAM where a special build is required. Ask support@iri.com for this in advance.
4. When writing job scripts that use multiple input files, use /INREC formatting to create a common record that can be processed. This common record maps common fields from the inputs so that only these fields can then be processed.
5. /INFILE and /INREC sections can both be specified in minimal terms to identify only those fields needed for processing and/or output display purposes; as with selection criteria, this method can be used to reduce data transformation bulk.
6. You can also use /INREC to perform pre-action field manipulations, such as a sub-string function to define a smaller sort key or save another processing pass.
7. Use wildcards with /INFILE for multiple input files with the same layout.
8. When specifying joins, use an /ALIAS for each input file to save time/space.
9. Use the /PREDICATE feature or an automation tool like Excel with formulas to repeat strings like /FIELD, POSITION=, SIZE= and to increment offsets. Paste field columns into notepad to build a text file you save with a .DDF extension.
10. Automation also precludes the need for abbreviations, which although still valid and can save time, can reduce script readability.

Using CoSort with Third-Party Tools

1. Learn how and why to use CoSort alongside legacy ETL tools like DataStage, Informatica, Pentaho and Talend (hint: to accelerate their slower transform jobs).
2. Use can also use CoSort SortCL jobs to accelerate multiple BI and analytic tools, including BIRT, Cognos, KNIME, Microstrategy, Oracle DV/D, Power BI, QlikView, R, Spotfire and Tableau.
3. It is possible to leverage third-party metadata through API-level hooks to SortCL.ddf layout files using erwin EDGE or Mapping Manager and MIMB. Using either platform to convert that proprietary metadata to ours saves time in re-platforming jobs.

4. Using CoSort sort drop-in replacement libraries the sorts in Micro Focus and AcuCOBOL, Software AG Natural, SAS 8, Unix /bin/sort, and the DB2 UDB load utility can save time and effort in upgrading their native sort verb performance.
5. For all IRI drop-in sort replacements, request the same bit architecture libraries that match both the CoSort and third-party package you have co-installed.
6. Some CoSort (SortCL) data mapping and masking jobs can also run in Hadoop without re-coding having to recode them, but require a premium Voracity license.

Using CoSort for Custom Processing

1. CoSort allows custom I/O via the SortCL commands /INPROCEDURE and /OUTPROCEDURE. Refer to the API chapter of the user guide.
2. Custom compares are allowed via /KEYPROCEDURE, custom key compares, and /ALTSEQ.
3. Custom data type conversions are also permitted in the /FIELD statement, allowing you to convert field values with your own functions.
4. SortCL also provides or supports custom field functions (for complex transformations, data quality, and protections like AES-256 encryption).

Using CoSort APIs and Compiling with CoSort Libraries

1. CoSort includes two callable APIs: cosort_r() and sortcl_routine(). If you are upgrading your API call from an earlier version of CoSort to Version 10, let us know; there were some adjustments made to accommodate frame characters.
2. Both APIs allow the use of the custom processing mentioned above.
3. Consider writing a wrapper around CoSort API calls, so if there are any changes to the APIs, necessary changes will be restricted to the contents of the wrapper.
4. The API allows passing of data into CoSort using files as well as a buffer structure. There are detailed examples in the User Guide explaining both.
5. While using shared libraries on Unix or Linux, make sure that the shared library path is updated to reflect the directory name where the CoSort shared libraries are stored.
6. When processing variable-length records, CoSort requires that each record be prepended by a short integer that stores the length of the record. CoSort provides the functions *getushort* and *putushort* for specific system architectures for use when writing API code.
7. The following are external libraries that may be required when linking: *-lpthread –lrt –lm –ldl*. These may differ slightly from platform to platform, so ensure that they are available in order to avoid linking errors.
8. While compiling with third-party libraries, verify that the bit architecture for CoSort matches the bit architecture of the embedding application.

Job Auditing and Error Reporting

1. Set a /STATISTICS=file statement in any SortCL jobs to produce runtime statistics. You can also create a /AUDIT=file to produce an XML report. Set /MONITOR_LEVEL=1 or above (up to 9) to send event messages to the console at runtime.
2. See this page for information about logging other IRI jobs beyond CoSort.
3. Errors are sent to stdout and the $COSORT_HOME or local directory .cserrlog.
4. Copy and paste the content or the error log and any errors that appear in the IRI Workbench problem tab.
5. Error messages are documented in Section E of the CoSort manual appendix.
6. Collect and submit help details per this web page or email support@iri.com. IRI will need: a) the items from 3-4 along with; b) the failing SortCL job script; c) your current performance settings displayed from the $COSORT_HOME/sortcl /rc command; d) any on-screen warning or error messages, e) the smallest failing subset of your input data; and, f) any observations related to similar data or scripts that were working -- and environmental conditions on the computer -- that are different between successful and unsuccessful runtimes.

Other Web and IRI Resources for Help and Training

1. The CoSort installation guide and user manual are in .pdf format, and are made available during NDA-confidential software downloads from the IRI FTP site. Documentation and advanced SortCL examples are also available on request, and are sometimes featured in the blog or newsletters. Those entitled can also request copies of past, current, or beta release manuals from their IRI agent.
2. Inside Workbench, clicking F1 will open context sensitive help. Additionally see help pages under *IRI Workbench Tools User Guides* under the Help Contents.
3. See self-learning articles at https://www.iri.com/services/training/courseware.
4. IRI also maintains public, non-expiring group pages on LinkedIn starting here.
5. Multiple how-to-videos are also posted to the IRI YouTube channel here.
6. Professional implementation and customization services -- as well as advanced training in the use of the CoSort Workbench, SortCL scripting language, CoSort API, tuning, and tuning -- are available from IRI and selected IRI sales and support offices worldwide at prevailing industry day rates. Bespoke services and ad hoc training courses are available in both remote and on-site engagement models for your convenience.

IRI
Total Data Management