Using ivy csv file utilities to program workflows

2017-01-13

Allart Ian Vogelesang ian.vogelesang@hds.com +1 408 396 6511

What is an ivy workflow?

HITACHI
Inspire the Next

- An ivy workflow is when you perform a test step using the ivy engine and then you examine what happened and decide what to do next.
- .ivyscript is the ivy workflow programming language
- .ivyscript itself looks a lot like C/C++, but it also has
  - Ivy engine control statements `[CreateRollup] "port" [quantity] 16;`
  - Builtin functions to access a csv file with a column title header row as if it were a spreadsheet by row and column.
  - These builtin csv file access utilities are also provided in the form of standalone command line executables for use outside ivy.

Live access to ivy data structures

HITACHI
Inspire the Next

- The only code that has access to ivy in-memory data structures while a subinterval sequence or "test step" is running is the dynamic feedback control mechanism.
  - Runs at end of subinterval after all test host and subsystem data is in
  - Can send out real-time workload parameter updates to rollup instances

Flexibility + Simplicity

HITACHI
Inspire the Next

- Provide the maximum flexibility for scripting ivy workflows with a modest investment of programming time
- Combination of two parts
  - Write out all ivy data structures as csv files with a column title (header) row
  - Provide the scripting language with
    - Scripting interpreter built-in functions to retrieve handy things to build csv file name, things like overall test folder names, or the name of the folder for the most recent test step*.
    - "spreadsheet-type" accessor functions that can also use column titles ("Overall IOPS") to select the column

* corresponding "get" and "set" methods to be added to ivy C++ API

.ivyscript builtin functions

HITACHI
Inspire the Next

- `string outputFolderRoot();` — from `[OutputFolderRoot]` statement – default "."
- `string testName();` — root part of ivyscript file without `.ivyscript` suffix
- `string masterlogfile();` — you can `log(masterlogfile(), "message\n");`
- `string testFolder();` — root folder for output from this run
- `string stepNNNN();` — from most recent [Go!], e.g. step0002
- `string stepName();` — from most recent [Go]
- `string stepFolder();` — subfolder for most recent [go] within testFolder()
- `string last_result();` — for most recent [Go], returns "`success`" or "`failure`"
- `string show_rollup_structure();` shows type / instance / workload thread hierarchy.

Corresponding "get" and "set" methods to be added to ivy C++ API

ivy csv file numbering – the header row is -1

HITACHI
Inspire the Next

Header row is row -1

Row 0 is test step 0 or subinterval 0

Use column number from 0, or say "Overall IOPS"

Test step csv files (not shown) have one line per subinterval (both host & subsystem data)

Summary csv files like this one have one line per test step.

	A	B	C	Q	R	S	AW				
1	Test Name	Step Numbe	Step Name	iogenerator type	blocksize	maxTags	Overall IOPS	Overall Decimal MB/s	Overall Average Blocksize (KiB)	Overall Little's Law Avg Q	Overall Average Service Time (ms)
2	demo9	step0000	iops_max	random_independent	4 KiB	32	2597.16	10.638	4	64.0024	24.6432
3	demo9	step0001	baseline_service_time	random_independent	4 KiB	32	25.81	0.105718	4	0.164588	6.37691
4	demo9	step0002	1.125_x_baseline	random_independent	4 KiB	32	555.849	2.27676	4	4.08014	7.34037
5	demo9	step0003	1.25_x_baseline	random_independent	4 KiB	32	1097.86	4.49682	4	8.74646	7.96686
6	demo9	step0004	1.5_x_baseline	random_independent	4 KiB	32	1486.01	6.08671	4	14.2187	9.56836
7	demo9	step0005	1.75_x_baseline	random_independent	4 KiB	32	1722.35	7.05476	4	19.2298	11.1649
8	demo9	step0006	2_x_baseline	random_independent	4 KiB	32	1898.07	7.77448	4	24.2199	12.7603
9	demo9	step0007	3_x_baseline	random_independent	4 KiB	32	2353.68	9.64067	4	45.0346	19.1337
10	demo9	step0008	4_x_baseline	random_independent	4 KiB	32	2602.77	10.6609	4	63.9994	24.589
11	demo9	step0009	5_x_baseline	random_independent	4 KiB	32	2601.7	10.6566	4	63.9962	24.5978

.ivyscript builtins vs. command line utilities

HITACHI
Inspire the Next

- The .ivyscript interpreter has a csvfile object type, and gives you functions to load a csv file into the set of currently-loaded csv files, examine the most recently set csvfile and to drop a csv file when you are finished with it, if it takes a lot of memory and it matters.
  - This optimizes for low overhead of making multiple accesses to a single csv file.
- The standalone command line utilities on each invocation load a csv file object with the target csv file in order to extract what was asked for.

.ivyscript csv file builtin functions 1/3

HITACHI
Inspire the Next

- `set_csvfile(string filename);`
  - Loads csv file into a kind of spreadsheet object, if it's not already loaded into memory.
  - You can load multiple csv files and switch back and forth.
  - All subsequent csvfile calls refer to the currently set csvfile.
- `drop_csvfile(string filename);`
  - If you are done with it and you would like to release the space.
- `int csvfile_rows();`
  - Number of rows following the header row.
  - Returns -1 if invalid file or file empty. Returns 0 if there was only a header row.
- `int csvfile_columns_in_row(int row);`
  `int csvfile_header_columns();` same as `csvfile_columns_in_row(-1)`

.ivyscript csv file builtin functions 2/3 – individual cells

HITACHI
Inspire the Next

- `string csvfile_cell_value(int row, int column);`
  `string csvfile_cell_value(int row, string column_header_text);`
  - You can refer to a column using an int, the column index from zero.
  - You can refer to a column using a string, the column header text.

- `string csvfile_raw_cell_value(int row, int column);`
  `string csvfile_raw_cell_value(int row, string column_header_text);`
  - ivy "wraps" text fields as a formula with a string constant, e.g. `="horse"`
    - This stops Excel from interpreting 1-1 as January 1st, and 00:00 from interpreting as a time.
  - The csv file functions normally "unwrap" csv column values, removing this kind of wrapper or removing simple double quotes surrounding a value, to treat `="horse"`, `"horse"` and `horse` the same
  - Retrieving the raw value give you exactly what was between the commas in the csv file.

.ivyscript csv file builtin functions 3/3 – headers & slices

HITACHI
Inspire the Next

- `string csvfile_column_header(int col);`
  - Give you the text of the column header

- `string csvfile_column(int col);`
  `string csvfile_column(string column_header);`
  - Gives you a "column slice" of the spreadsheet showing "raw" values.
  - E.g. `"IOPS,55,66,55,44"`
  - Demo number 9 shows iterating through the column slices to write out the transpose of a csv file.

- `string csvfile_row(int row);`
  - Gives you a "row slice" of the spreadsheet showing the "raw" values.
  - E.g. `="random_independent",="4 KiB",32,2601.7`

Command line executables – ivy csv file utilities

HITACHI
Inspire the Next

- To make it easy to navigate ivy output csv files from other scripting or programming languages, the .ivyscript csv utilities are also provided in the form of standalone command line executables.

- Note: when you specify command line arguments, it can be a pain with bash if you are dynamically generating commands to be executed if any of the command line arguments have spaces in them.
  - Fortunately, ivy treats `"Overall IOPS"` as equivalent to `overall_iops`

ivy's use of Excel formula format values

HITACHI
Inspire the Next

- In order to prevent Excel from interpreting a cell according to its value, for example to prevent Excel from storing the cell value as the date January 1st when the csv file says "1-1", ivy uses the trick to show a character string as a formula, as in ="1-1".

- In these utility functions, when you retrieve using "csv_cell_value", any ivy formula wrapper is removed, and quoted strings are unpacked, meaning the quotes are removed and escaped characters are rehydrated.

- The "csv_raw_cell_value" executable gives you the exact original text between the commas in the csv file.

- If something goes wrong, each of the ivy csv utility executables will say something starting with `<Error>`.

Command line ivy csv file utilities 1/4

HITACHI
Inspire the Next

- `csv_lookup_column` filename column_header
  - returns the column number for the specified column title (column header).
  - e.g. `csv_lookup_column x.csv "Overall IOPS"`
- `csv_column_header` filename column_number
  - prints the column header for the specified column number
  - e.g. `csv_column_header x.csv 0`
- `csv_rows` filename
  - prints the number of data rows following the header row.
  - prints 0 if there is only a header row.
  - e.g. `csv_rows x.csv`

Command line ivy csv file utilities 2/4

HITACHI
Inspire the Next

- `csv_header_columns` filename
  - prints the number of columns (number of un-quoted commas plus one) in the header row
  - e.g. `csv_header_columns /home/user/Desktop/a.csv`
- `csv_columns_in_row` filename row_number
  - prints the number of columns (number of un-quoted commas plus one) in the specified row
  - e.g. `csv_columns_in_row x.csv 3`
- `csv_raw_cell_value` filename row_number column_number
  `csv_raw_cell_value` filename row_number column_header
  - prints the exact characters found between commas in the csv file
  - e.g. `csv_raw_cell_value x.csv 4 5`
    e.g. `csv_raw_cell_value x.csv 4 "Overall IOPS"`

Command line ivy csv file utilities 3/4

HITACHI
Inspire the Next

- `csv_cell_value` filename row_number column_number
  `csv_cell_value` filename row_number column_header
  - e.g. `csv_cell_value x.csv 4 5`
    e.g. `csv_cell_value x.csv 4 "Overall IOPS"`
  - This "unwraps" the raw cell value.
  - First if it starts with =" and ends with ", we just clip those off and return the rest.
  - Otherwise, an unwrapped CSV column value first has leading/trailing whitespace removed and then if what remains is a quoted string, the quotes are removed and any internal "escaped" quotes have their escaping backslashes removed.

Command line ivy csv file utilities 4/4

HITACHI
Inspire the Next

- `csv_row` filename row_number
  - prints the entire row
  - e.g `csv_row x.csv 2`

- `csv_column` filename column_number
  `csv_column` filename column_header
  - prints column slice
  - e.g. `csv_column x.csv 4`
    e.g. `csv_column x.csv "Overall IOPS"`

Demo csv file with retrieval script provided

HITACHI
Inspire the Next

- See `test_csv.csv`, `test_csv.sh`, and `test_csv.sh.output.txt` in the doc subfolder at https://github.com/Hitachi-Data-Systems/ivy.

HITACHI

Inspire the Next