| LCOV - code coverage report |
| --- |
|  |
| | Current view: | [top level](http://docs.google.com/index.html) - [FixedModeFlow/src](http://docs.google.com/index.html) - FixedModeCellProcessor.cpp (source / [functions](http://docs.google.com/FixedModeCellProcessor.cpp.func.html)) |  |  | Hit | Total | Coverage | | --- | --- | --- | --- | --- | --- | --- | | Test: | EBM9K Converter Coverage Report |  | Lines: | 76 | 110 | 69.1 % | | Date: | 2012-03-23 |  | Functions: | 11 | 12 | 91.7 % | | Legend: | Lines: hit not hit | Branches: + taken - not taken # not executed |  | Branches: | 17 | 44 | 38.6 % | |  | |
|  |

|  |
| --- |
| Branch data Line data Source code  1 : : #include "ConverterParams.h"  2 : : #include "ErrorHandler.h"  3 : : #include "FixedModeCellProcessor.h"  4 : : #include "pattern.h"  5 : : #include "PatternProcessor.h"  6 : : #include "SlidingWindow.h"  7 : : #include "SDDB.h"   8 : : #include "print.h"  9 : : #include "HealthCheckCounter.h"  10 : :   11 : : void sigHandler(int i)  12 : 0 : {  13 : 0 : nft::e9k\_error("Received signal %d\n", i);  14 : : }  15 : :   16 : : namespace nft {  17 : :   18 : : void  19 : 1244 : FixedModeCellProcessor::scheduleCell(Cell::Pointer inCell) {  20 : 1244 : mCells.scheduleCell( inCell );  21 : : }  22 : :   23 : : //This is used by scheduler to notify the cell processor,  24 : : //that all cells have been scheduled.  25 : : void  26 : 1066 : FixedModeCellProcessor::notifyCellSchedulingCompletion() {  27 : 1066 : boost::mutex::scoped\_lock l( mMutex );  28 : 1066 : mEventOccured.notify\_one();  29 : : }  30 : :   31 : : void  32 : 1096 : FixedModeCellProcessor::stop() {  33 : 1096 : boost::mutex::scoped\_lock l( mMutex );  34 : 1096 : mStop = true;  35 : 1096 : mEventOccured.notify\_one();  36 : : }  37 : :   38 : : void  39 : 1096 : FixedModeCellProcessor::operator()() throw() {  40 : : //signal(11, sigHandler);  41 : : //signal(6, sigHandler);  42 : : //signal(8, sigHandler);  43 : 1096 : SDDB \*pSDDB = SDDB::instance();  44 : 2286 : while( true )  45 : : {  46 : 2286 : HealthCheckCounter::instance()->incrCounter(  47 : : HealthCheckCounter::eUpdateFrequent);  48 : : {  49 : 2286 : boost::mutex::scoped\_lock l( mMutex );  50 : : //while( !mStop && mCells.isEmpty() )  51 [ + - ][ + + ]: 2286 : if( !mStop && mCells.isEmpty() )  52 : : {  53 : 2113 : mEventOccured.wait( l );  54 : : }  55 [ + + ]: 2286 : if( mStop )  56 : : {  57 : 1048 : try  58 : : {  59 : 1048 : pSDDB->allCellProcessingCompleted();  60 : : }  61 [ # # ]: 0 : catch( DPL::Error \*pChild ) {  62 : 0 : ErrorHandler \*pErr = ErrorHandler::instance();  63 : 0 : pErr->errorAdd(pChild, \_\_FILE\_\_, \_\_FUNCTION\_\_, \_\_LINE\_\_,  64 : : ErrorHandler::eErrFunction,  65 : : pErr->getErrorMsg(  66 : : ErrorHandler::eErrFunction,  67 : : "SDDB::allCellProcessingCompleted"),  68 : : getThreadManager());  69 : 0 : return;  70 : : }  71 : :   72 : 1048 : return;  73 : : }  74 : : }  75 : : //Empty check is not required here  76 : 1238 : Cell::Pointer cell = mCells.getNextCell();  77 : 1238 : e9k\_debug("Found a new cell\n");  78 [ - + ]: 1238 : if( NULL == cell )  79 : : {  80 : 1238 : continue;  81 : : }  82 : 1238 : int nX = 0, nY = 0, eX = 0, eY = 0;  83 : 1238 : GroupContext tmpPGC = \*(cell->pgc);  84 : : //PatternContext tmpPPC = tmpPGC.ppc;  85 : : //tmpPGC.ppc = tmpPPC;  86 : :   87 : 1238 : try   88 : : {  89 : 1238 : pSDDB->CellAvailable(cell);  90 : : }  91 [ # # ]: 0 : catch( DPL::Error \*pChild ) {  92 : 0 : ErrorHandler \*pErr = ErrorHandler::instance();  93 : 0 : pErr->errorAdd(pChild, \_\_FILE\_\_, \_\_FUNCTION\_\_, \_\_LINE\_\_,  94 : : ErrorHandler::eErrFunction,  95 : : pErr->getErrorMsg(  96 : : ErrorHandler::eErrFunction,  97 : : "SDDB::CellAvailable"),  98 : : getThreadManager());  99 : 0 : return;  100 : : }  101 : :   102 : :   103 : 1238 : std::vector<PatternPosition> positions;  104 : 7119 : while(true)  105 : : {  106 : 7119 : EP newEP;  107 : 7119 : try  108 : : {  109 [ + + ]: 7119 : if( -1 == vsbGetEP( &tmpPGC, &newEP, &nX, &nY, &eX, &eY ) )  110 : : {  111 : 1190 : break;  112 : : }  113 : : }  114 [ + - ]: 32 : catch(DPL::Error \*pChild)  115 : : {  116 : 32 : ErrorHandler \*pErr = ErrorHandler::instance();  117 : 32 : pErr->errorAdd(pChild, \_\_FILE\_\_, \_\_FUNCTION\_\_, \_\_LINE\_\_,  118 : : ErrorHandler::eErrFunction,  119 : : pErr->getErrorMsg( ErrorHandler::eErrFunction,  120 : : "ConversionEngine::convertFixedMode"),  121 : : getThreadManager());  122 : 32 : return;  123 : : }  124 : 5897 : positions.clear();  125 : 5897 : ParsedPattern pat;  126 : : //(\*(cell->pgc)).ppc = tmpPGC.ppc;  127 : 5897 : try  128 : : {  129 : 5897 : mExploder.getParsedPattern(  130 : : &tmpPGC, positions, newEP, nX, nY, eX, eY,  131 : : cell->startPos, pat );  132 : : }  133 [ # # ]: 0 : catch( DPL::Error \*pChild )  134 : : {  135 : 0 : ErrorHandler \*pErr = ErrorHandler::instance();  136 : 0 : pErr->errorAdd(pChild, \_\_FILE\_\_, \_\_FUNCTION\_\_, \_\_LINE\_\_,  137 : : ErrorHandler::eErrFunction,  138 : : pErr->getErrorMsg(  139 : : ErrorHandler::eErrFunction,  140 : : "PatternExploder::getParsedPattern"),  141 : : getThreadManager() );  142 : 0 : return;  143 : : }  144 : 5897 : try  145 : : {  146 : 5897 : pSDDB->EPGenerationCompleted(positions.size());  147 : : }  148 [ # # ]: 0 : catch( DPL::Error \*pChild )  149 : : {  150 : 0 : ErrorHandler \*pErr = ErrorHandler::instance();  151 : 0 : pErr->errorAdd(pChild, \_\_FILE\_\_, \_\_FUNCTION\_\_, \_\_LINE\_\_,  152 : : ErrorHandler::eErrFunction,  153 : : pErr->getErrorMsg(ErrorHandler::eErrFunction,  154 : : "SDDB::EPGenerationCompleted"),  155 : : getThreadManager());  156 : 0 : return;  157 : : }  158 : :   159 : :   160 : 5897 : try   161 : : {  162 : 5897 : pSDDB->ParsedPatternGenerated(cell, pat, positions);  163 : : }  164 [ # # ]: 0 : catch( DPL::Error \*pChild )   165 : : {  166 : 0 : ErrorHandler \*pErr = ErrorHandler::instance();  167 : 0 : pErr->errorAdd(pChild, \_\_FILE\_\_, \_\_FUNCTION\_\_, \_\_LINE\_\_,  168 : : ErrorHandler::eErrFunction,  169 : : pErr->getErrorMsg(  170 : : ErrorHandler::eErrFunction,  171 : : "SDDB::ParsedPatternGenerated"),  172 : : getThreadManager());  173 : 0 : return;  174 : : }  175 : :   176 : : uint32\_t numLayers  177 : 5897 : = mParams->getStartupMessage()->uiSFLCountInDpb;  178 : 5897 : for(std::vector<PatternPosition>::iterator it = positions.begin();  179 : : it != positions.end(); ++it )  180 : : {  181 : 7759 : pat.setPosition( \*it );  182 [ + - ][ + + ]: 15680 : for( uint32\_t i = 0; i < numLayers; i++ )  183 : : {  184 : 7937 : pSDDB->setLayerNumber(i);  185 : :   186 : 7937 : try  187 : : {  188 : 7937 : pSDDB->ParsedPatternReadyForProcessing(pat);  189 : : }  190 [ # # ]: 0 : catch( DPL::Error \*pChild )  191 : : {  192 : 0 : ErrorHandler \*pErr = ErrorHandler::instance();  193 : 0 : pErr->errorAdd(pChild, \_\_FILE\_\_, \_\_FUNCTION\_\_, \_\_LINE\_\_,  194 : : ErrorHandler::eErrFunction,  195 : : pErr->getErrorMsg(  196 : : ErrorHandler::eErrFunction,  197 : : "SDDB::ParsedPatternReadyForProcessing"),  198 : : getThreadManager());  199 : 0 : return;  200 : : }  201 : :   202 : 7937 : try  203 : : {  204 : 7937 : mPatternProcessor->process( pat,i );  205 : : }  206 [ + - ]: 16 : catch( DPL::Error \*pChild )  207 : : {  208 : 16 : ErrorHandler \*pErr = ErrorHandler::instance();  209 : 16 : pErr->errorAdd(pChild, \_\_FILE\_\_, \_\_FUNCTION\_\_, \_\_LINE\_\_,  210 : : ErrorHandler::eErrFunction,  211 : : pErr->getErrorMsg(  212 : : ErrorHandler::eErrFunction,  213 : : "PatternProcessor::process"),  214 : : getThreadManager() );  215 : 16 : return;  216 : : }  217 : 7921 : try  218 : : {  219 : 7921 : pSDDB->ParsedPatternProcessingCompleted();  220 : : }  221 [ # # ]: 0 : catch( DPL::Error \*pChild )  222 : : {  223 : 0 : ErrorHandler \*pErr = ErrorHandler::instance();  224 : 0 : pErr->errorAdd(pChild, \_\_FILE\_\_, \_\_FUNCTION\_\_, \_\_LINE\_\_,  225 : : ErrorHandler::eErrFunction,  226 : : pErr->getErrorMsg(  227 : : ErrorHandler::eErrFunction,  228 : : "SDDB::ParsedPatternProcessingCompleted"),  229 : : getThreadManager());  230 : 0 : return;  231 : : }  232 : :   233 : : }  234 : : }  235 : : }  236 : 1190 : try  237 : : {  238 : 1190 : pSDDB->cellProcessingCompleted();   239 : : }  240 [ # # ]: 0 : catch( DPL::Error \*pChild )  241 : : {  242 : 0 : ErrorHandler \*pErr = ErrorHandler::instance();  243 : 0 : pErr->errorAdd(pChild, \_\_FILE\_\_, \_\_FUNCTION\_\_, \_\_LINE\_\_,  244 : : ErrorHandler::eErrFunction,  245 : : pErr->getErrorMsg(  246 : : ErrorHandler::eErrFunction,  247 : : "SDDB::cellProcessingCompleted"),  248 : : getThreadManager());  249 : 0 : return;  250 : : }  251 : :   252 [ + - ]: 1190 : delete cell;  253 [ + + ]: 1190 : if( mCells.isEmpty() )  254 : : {  255 : 1018 : mWindow->notifyProcessingCompletion();  256 : : }  257 : : }  258 : : }  259 : :   260 : :   261 : : FixedModeCellProcessor::FixedModeCellProcessor(  262 : : ThreadID inThreadID, ConverterParams \*inParams,  263 : : PatternProcessor \*inPatternProc )  264 : 1096 : :Thread(),  265 : : mParams( inParams ),  266 : : mExploder(),  267 : : mPatternProcessor( inPatternProc ),  268 : : mStop(),  269 : : mMutex(),  270 : : mEventOccured(),  271 : 2192 : mWindow() {  272 : 1096 : setThreadID( inThreadID );  273 : : }  274 : :   275 : 2192 : FixedModeCellProcessor::~FixedModeCellProcessor() {  276 [ + - ][ # # ]: 1096 : delete mPatternProcessor;  [ # # ]  277 : 1096 : }  278 : :   279 : : void  280 : 1096 : FixedModeCellProcessor::setWindow(SlidingWindow \* value) {  281 : 1096 : mWindow = value;  282 : : }  283 : :   284 : :   285 : 1215 : } // namespace nft |

|  |
| --- |
| Generated by: [LCOV version 1.9](http://ltp.sourceforge.net/coverage/lcov.php) |