PROCESS-STRIPED BUFFERING WITH GEN_STREAM

A NEW BEHAVIOUR PROPOSED FOR R15A

JAY NELSON

HTTP://WWW.DUOMARK.COM/

@DUOMARK

GENESIS

- □ WIDEFINDER (TIM BRAY'S* CONCURRENCY CHALLENGE)
    - □ COUNT WEBPAGE VISIT FREQUENCY
    - □ ~10 LINES OF RUBY
- □ WANTED TO SCALE TO MULTI-CORE WITHOUT EFFORT
- □ *HTTP://WWW.TBRAY.ORG/ONGOING/WHEN/200X/2007/09/20/WIDE-FINDER

WIDEFINDER RESULTS

- □ ERLANG FARED POORLY (INITIALLY)
  - □ TEXT I/O PERFORMANCE WAS LACKING
- □ CONCERTED EFFORT BY ERLANGERS
  - □ RESULT = OVER 350 LINES OF CODE
  - □ (VINOSKI, CAOYUAN AND OTHERS)

COMMON PATTERN

- □ PATTERN EMERGED IN ERLANG SUBMISSIONS
  - □ BINARY READ FILE
  - □ FIND LINE BREAKS
  - □ DISTRIBUTE LINES
- □ SEEMED SIMPLE, INVOLVED HUNDREDS OF SLOC
  - □ MIRRORED MY EARLIER EXPERIMENTS WITH BINARIES
  - □ CAN'T ASSUME BINARY FITS IN MEMORY

WIDEFINDER2 (ASIDE)

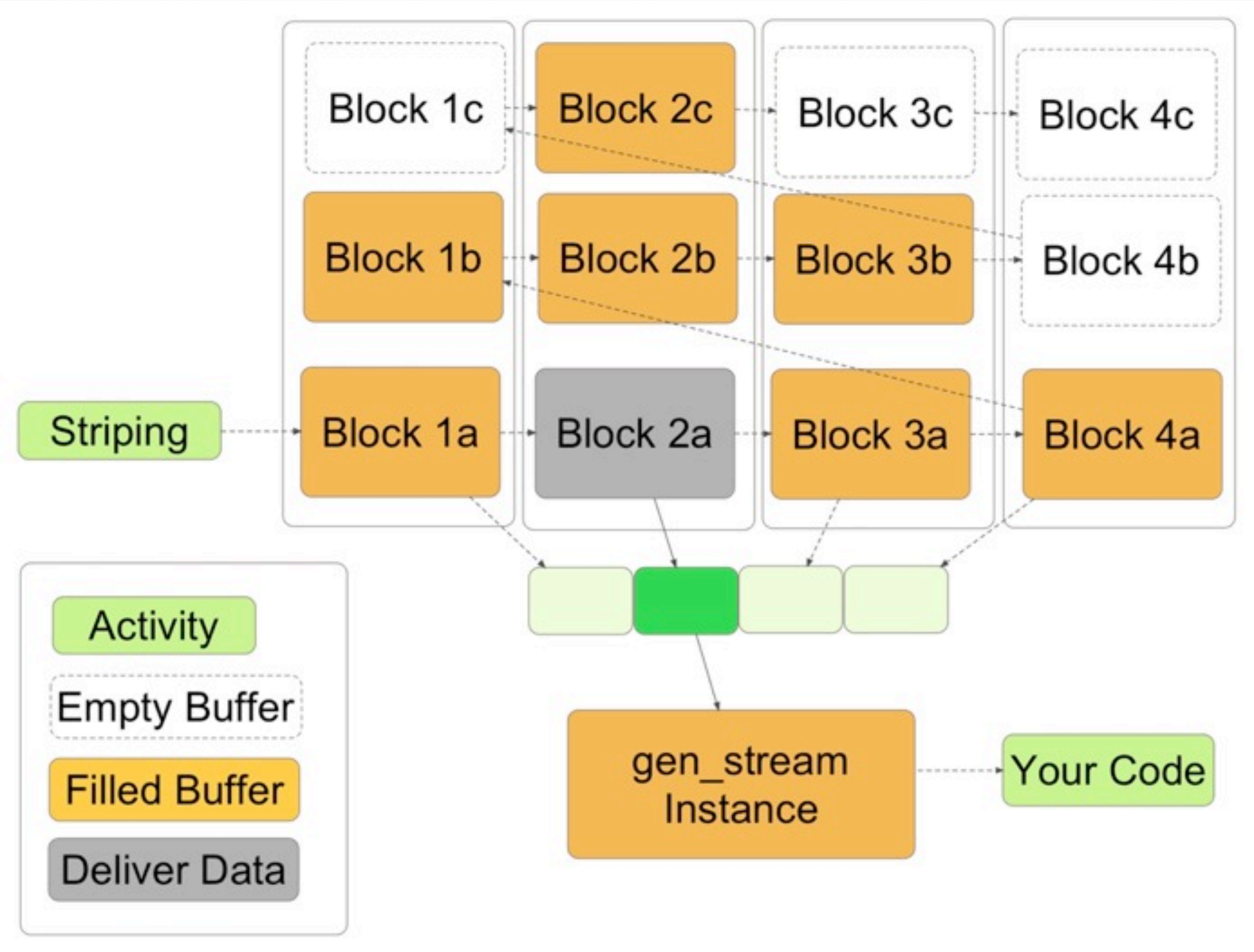
- □ FINAL WIDEFINDER2 SOLUTIONS ARE MOSTLY C
- □ ULTIMATE WINNER OF WIDEFINDER2
    - □ HTTP://WWW.1024CORES.NET/
    - □ BLOG IS A GOOD READ ON CONCURRENCY ISSUES

GEN_STREAM

CONCEPT

- □ BUILT ON GEN_SERVER
- □ MAINTAINS AN INTERNAL “MATRIX” OF BUFFERS
  - □ EACH COLUMN IS A PROCESS
  - □ EACH CELL IS A “BLOCK” OF MEMORY
- □ SERIAL STREAM IS STRIPED ACROSS PROCESSES
  - □ ADJACENT SEGMENTS ARE IN DIFFERENT PROCESSES
  - □ COLUMN REFILLS INTERLEAVE WITH REQUESTS

CONCEPT (CONT.)



EXAMPLE API USAGE

```
{ok, Pid} =
  gen_stream:start_link([{stream_type,
                          {binary, BinaryInMemory},
                          {num_procs, 4},
                          {chunks_per_proc, 3}]);
read_all(Pid).

read_all(Pid) ->
  case gen_stream:next_block(Pid) of
    {block, Block} ->
      process_block(Block),
      read_all(Pid);
    {end_of_stream} ->
      gen_stream:stop(Pid)
  end.
```

IMPLEMENTATION

- START / START_LINK STREAM_TYPE OPTIONS (REQ'D)
  - `{stream_type, {binary, Bin::binary()}}`
  - `{stream_type, {file, FileName::string()}}`
  - `{stream_type, {behaviour, Mod::atom(), ModArgs::list()}}`
- DETERMINES SOURCE DATA TYPE
  - BUILT-INS USE SUB-BINARIES WHERE POSSIBLE

IMPLEMENTATION (CONT.)

- □ START / START_LINK BUFFER SIZING OPTIONS
  - □ {num_procs, pos_integer()} => concurrency
  - □ {chunks_per_proc, pos_integer()} => stacked buffers
  - □ {chunk_size, pos_integer()} => single buffer size
  - □ {block_factor, pos_integer()} => # records per buffer
- □ LIMIT MAXIMUM MEMORY USAGE
- □ ALLOW PACKING OF SMALL DATA
- □ DEFINE CONCURRENT DATA LOADING

IMPLEMENTATION (CONT.)

- START / START_LINK REPLAY OPTIONS
  - {is_circular, boolean()} => continuous data stream
- START / START_LINK TRANSFORM CHUNK OPTIONS
  - {x_mfa, {module(), atom(), list()}}
  - {x_fun, fun()}
  - CONVERTS DATA CONCURRENTLY AS IT IS LOADING

BEHAVIOUR INTERFACE

```
behaviour_info(callbacks) ->
    [
     {init, 1},                % Creates ModState
     {stream_size, 1},         % may be ‘is_circular’
     {inc_progress, 2},        % Seen + ThisChunkSize
     {extract_block, 5},
     {extract_final_block, 5},
     {terminate, 2},
     {code_change, 3}
    ];
```

EXTRACT_BLOCK/5

□ MODULE STATE (FROM MODULE:INIT() CALL)

□ POSITION (OFFSET FROM START OF STREAM)

□ NUMBER OF BYTES TO PRODUCE

□ CHUNK SIZE (NUMBER OF BYTES IN A CHUNK)

□ BLOCKING FACTOR (E.G., 10 CHUNKS PER BLOCK)

EXTRACT_FINAL_BLOCK/5

- SAME PARAMETERS AS EXTRACT_BLOCK/5
    - NUMBER OF BYTES IS CAPPED TO STREAM_SIZE
    - GEN_STREAM HANDLES CIRCULARITY

DYNAMICS

- INIT/1 - INSTANTIATES INTERNAL PROCESSES
  - SEND `{next_block, self()}` TO EACH PROCESS
- CLIENT REQUESTS `gen_stream:next_block(Pid)`
- CLIENT AND FILL BUFFER REQUESTS INTERLEAVE
- IF BUFFER EMPTY, CLIENT REQUEST IS IMMEDIATE FILL
  - FETCH, RETURN AND MESSAGE SELF TO FILL BUFFER

IMPLICATIONS

- □ ALWAYS REPRESENTS A SERIAL, ORDERED STREAM
- □ DESIGNED FOR PULL SEMANTICS (PUSH CAN OVERFLOW)
- □ EQUIVALENT TO A COMPREHENSION ON EXTERNAL DATA
  - □ CAN IMPLEMENT “INFINITE COMPREHENSIONS”
- □ MAIN CONCURRENCY IS OVERLAPPED DATA FETCHES
  - □ SECONDARY CONCURRENCY IN REFILLING BUFFERS
  - □ CONCURRENT “ON-THE-FLY” TRANSFORMATIONS

USER CHOICES

- STREAM DYNAMICS
  - RESOURCES CONSUMED: MEMORY, PROCESSES
- DATA PROCESSING MODEL
  - DATA GRANULARITY / ELEMENT BLOCKING
- ARCHITECTURAL CHOKE POINTS
  - THROTTLE DATA TIMING / THROUGHPUT
  - ADAPTIVELY CONTROLLED ON EACH INSTANTIATION

PROMISE
(HOPE?)

EFFICIENT TEXT FILES

- □ COVERS THE WIDEFINDER CODE EXAMPLES
  - □ BINARY BLOCKS OF TEXT
  - □ ALLOWS VARIABLE CHUNK SIZES
- □ USER-DEFINED x_mfa OR x_fun TO BREAK BLOCKS
  - □ OPTIONALLY ELIMINATE OR FILTER DATA BLOCKS
  - □ COULD ALSO COMPRESS / DECOMPRESS
  - □ ANY DATA TRANSFORMATION

FIXED-SIZE RECORDS

- □ EXTREMELY EFFICIENT FIXED LENGTH RECORD LOADING
  - □ PREDICTIVE LOCATIONS ALLOW FULL CONCURRENCY
  - □ DATA CAN FLOW THROUGH BUFFERS AS BINARIES
- □ INDEX GENERATION (RECORDS AND LOCATIONS)
- □ BULK-LOADING OF VERY SHORT RECORDS
  - □ block_factor LOADS MULTIPLE RECORDS PER CHUNK
  - □ TRANSFORM CAN SPLIT TO A LIST OF SUB-BINARIES

BUFFERING

- □ ORIGINAL GOAL OF THE PATTERN
- □ ON REFLECTION PROBABLY LEAST USEFUL FEATURE
  - □ I/O ALREADY BUFFERED AT LEAST TWICE
  - □ HTTP://SNA-PROJECTS.COM/KAFKA/DESIGN.PHP
  - □ FROM LINKEDIN'S KAFKA MESSAGING

STREAM IDIOM

- □ CONCISE, EASY-TO-USE INTERFACE
  - □ BINARY, FILE OR FUNCTIONAL GENERATION
  - □ (FUTURE CONTINUATION-BASED OPTION)
  - □ INFINITE DATA / LAZY DATA GENERATION
- □ STANDARDIZES ALGORITHMS TO "UNITS OF WORK"
  - □ ARCHITECTURAL LEVEL COMPREHENSIONS
  - □ EXTENDS MAPPING BEYOND MEMORY SIZE

SEQUENCING EVENTS

- □ STREAMS CAN BE SEQUENTIALLY ORDERED “EVENTS”
  - □ REPRODUCIBLE TESTING SCENARIOS
  - □ SCRIPTED EVENTS CAN DRIVE STATE MACHINES
  - □ SCRIPTING AS AN ARCHITECTURAL PATTERN
- □ POOLED SOURCE OF SLOW TO GENERATE SEQUENCES
  - □ BEWARE THAT NEXT_BLOCK MAY TIMEOUT

TESTING

- MEMORY EFFICIENT, REPEATABLE EVENTING
  - LARGE EXTERNAL SOURCE OF TEST EXAMPLES
  - GENERATED TEST CASES VIA A MODULE
  - INFINITE STREAMS OF DATA (CIRCULAR OR NOT)
  - INFINITE RANDOM SAMPLING FROM A SET
  - STRESS TESTING / MEMORY LEAK IDENTIFICATION
- DYNAMICALLY SCRIPTED EVENTING

FEEDBACK

- □ CODE IS COOKING IN 'PU' ON GITHUB: ERLANG/OTP
  - □ jn/gen_stream (stdlib) (730c7fd)
- □ WILL BE AVAILABLE AT HTTP://WWW.DUOMARK.COM/
  - □ EASIER TO LOAD FROM THE SHELL
- □ PLEASE TRY IT, GIVE FEEDBACK -- GOOD OR BAD
- □ DEMAND ACCEPTANCE FROM YOU SWEDISH OTP REP!!