**NAME**

**SDM library** - the SDM presents novel algorithms to efficiently predict block boundaries and a runtime system that enables efficient block-level parallel decompression. The SDM execution model features speculative pipelining with three stages: Scanner, Decompressor, and Merger.

**SDM\_bzip2** – parallel decompressor using SDM

**SYNOPSIS**

#include <papl.h>

struct PAPL conf;

conf.scan = bzip2\_scan;

conf.decompress = bzip2\_do\_work;

**DESCRIPTION**

The SDM API functions are classified into two categories. The first category includes functions already implemented by SDM to initiate and orchestrate pipeline execution. The second category defines interface functions that encapsulate algorithm-specific prediction and decompression codes and must be implemented by the programmer.

|  |  |
| --- | --- |
| **Operation** | **Description** |
| **System Functions** | |
| **init**(conf) | Read file size and create queue. If the commit mode is DIST, allocate some buffer |
| **destroy**(conf) | Destroy the created queue and buffers |
| **create\_process**(function, &pid, arg) | Create a new process with process id that will execute the function with arg |
| **join\_process**(conf) | Wait for worker processes with the configured number of processes |
| **produce**(dst, &val, sizeof(val)) | Enqueue val into software queue to send to dst; |
| **consume** (src, &val, sizeof(val)) | Dequeue val from src |
| **scan**(conf) | Execute scan process; Forward a speculative block boundary to targeted processes |
| **decompress**(decomp\_pid) | Execute decompress process; compute the compressed data with speculative boundary |
| **merge**(conf) | Execute merge process; write a decompressed data in output stream (Centralized commit) |
| **recover\_misspec**(real\_pos) | Handle recovery from mis-speculation |
| **insert\_merge\_list**(output\_buf, len, merge\_list) | Insert decompressed chunk in linked list; the decompressed chunk is stored in merge list for each process (Distributed commit) |
| **User Functions** | |
| next\_pos = **predict\_boundary**(conf.in\_file, pos, conf.chunk\_size, &eof) | Find the boundary of block in compressed stream by using prediction algorithm |
| **decompress\_chunk**(conf.in\_file, pos, &err, &real\_pos) | Decompress a given chunk and calculate the real pos |

Configuration parameters for SDM execution are defined in conf structure. **chunk size** specifies the amount of work dispatched to a decompressor process at a time in independently decompressible units (IDUs). The IDU of bzip2 is a compressed block; the IDU of gzip is a group of blocks beginning with a stored block; the IDU of H.264 is an IDR-frame group.

|  |  |  |  |
| --- | --- | --- | --- |
| **Parameter** | **Type** | **Default** | **Description** |
| process\_num | int | 4 | Number of decompressor processes |
| chunk\_size | int | 1 | Chunk size in independently decompressible units |
| commit\_mode | enum | CENT | Commit mode: CENT (centralized), DIST (distributed) |
| in\_file | char\* | Input file | Name of input file |
| out\_file | char\* | Output file | Name of output file |

**RETURN VALUES**

* next\_pos = **predict\_boundary**(conf.in\_file, pos, conf.chunk\_size, &eof) -

The **predict\_boundary** function returns the starting position of the next chunk, which is sent to both decompressor stage (for decompression) and merger stage (for misspeculation detection). The merger process is responsible for misspeculation detection and recovery. It compares the predicted **pos (next\_pos)** value received from the scanner process with the corresponding, non-speculative **pos** value received from a decompressor process.

If a misspeculation is detected, all speculative scanner and de- compressor processes are squashed, and the rest of the program is sequentially executed. It is our design decision not to restart parallel execution since the prediction algorithms

**AUTHORS**

Hakbeom Jang <hakbeom@.skku.edu>, Channoh Kim <channoh@skku.edu> and Jae W. LEE <[jaewlee@snu.ac.kr](mailto:jaewlee@snu.ac.kr)>

* [LCTES '13] Hakbeom Jang, Channoh Kim, and Jae W. Lee, "**practical speculative parallelization of variable-length decompression algorithms**", 14th ACM SIGPLAN/SIGBED International Conference on L nguages, Compilers, and Tools for Embedded Systems (LCTES), Seattle, Washington, June 2013. (Also appeared in ACM SIGPLAN Notices, 48(5), May 2013.)

Architecture and Code Optimization Lab, Seoul National University

**ACKNOWLEDGMENT**

The development of this package was supported by the IT R&D program of MKE/KEIT (No. 10041244, SmartTV 2.0 Software Platform).