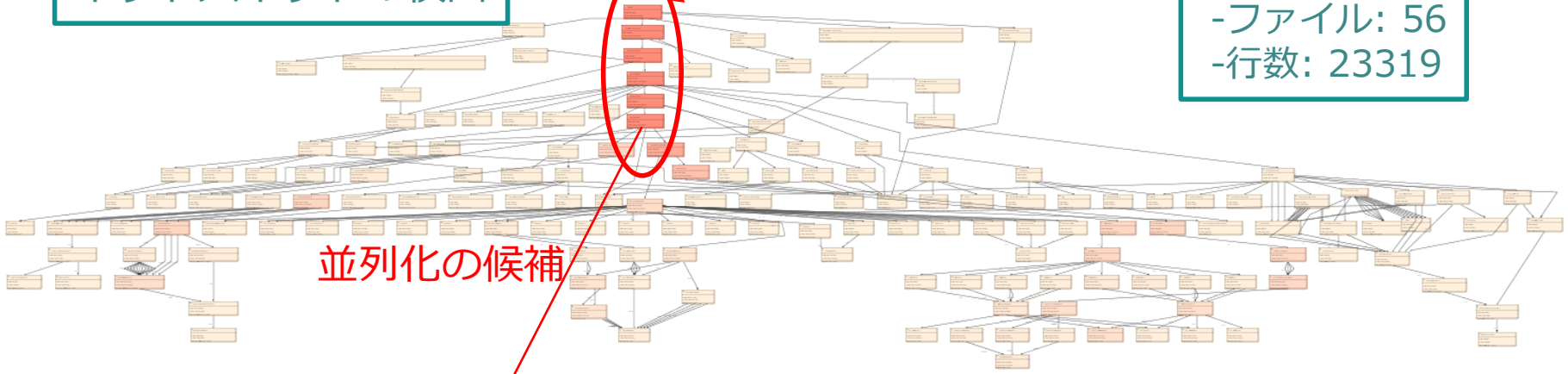
SLX Parallelizer – 動的・静的解析

ホットスポットの検出

ホットスポット関数

-関数: 162
-ファイル: 56
-行数: 23319



並列化

有益な情報の可視化!

各行の
プロファイル

並列タスクを
ハイライト

```
static int ParseFrame(VP8Decoder * const dec, VP8Io * io)
{
    int ret_val;
    for (dec->mb_y_ = 0; dec->mb_y_ < dec->br_mb_y_; ++dec->mb_y_)
    {
        // Parse bitstream for this row.
        VP8BitReader *const token_br = &dec->parts_[dec->mb_y_ & (dec->num_parts_ - 1)];
        if (!VP8ParseIntraModeRow(&dec->br_, dec))
        {
            ret_val = VP8SetError(dec, VP8_STATUS_NOT_ENOUGH_DATA, "Premature end-of-partition0 encountered.");
        }
        for (; dec->mb_x_ < dec->mb_w_; ++dec->mb_x_)
        {
            if (!VP8DecodeMB(dec, token_br))
            {
                ret_val = VP8SetError(dec, VP8_STATUS_NOT_ENOUGH_DATA, "Premature end-of-file encountered.");
            }
        }
        VP8InitScanline(dec);   // Prepare for next scanline

        // Reconstruct, filter and emit the row.
        if (!VP8ProcessRow(dec, io))
        {
            ret_val = VP8SetError(dec, VP8_STATUS_USER_ABORT, "Output aborted.");
        }
    }
    if (dec->mt_method_ > 0)
    {
        if (!WebPGetWorkerInterface()->Sync(&dec->worker_))
            ret_val = 0;
    }
```

```
static int ParseFrame(VP8Decoder * const dec, VP8Io * io)
{
    int ret_val;

Multiple markers at this line
  - This loop only present PLP parallelism pattern with an speedup of 1.30369
  - PLP - Recommended pipeline configuration has 2 stages. Estimated speedup is: Local 1.30369, Global 1.30153
  - PLP - Pipeline with 2 stages has estimated speedup of 1.30369. Efficiency: 0.651845
  - DLP - Loop does not provide DLP because of loop carried dependency on variable Heap5_WebPSafeMalloc_224 [WAR]
        {
            ret_val = VP8SetError(dec, VP8_STATUS_NOT_ENOUGH_DATA, "Premature end-of-partition0 encountered.");
        }
        for (; dec->mb_x_ < dec->mb_w_; ++dec->mb_x_)
        {
            if (!VP8DecodeMB(dec, token_br))
            {
                ret_val = VP8SetError(dec, VP8_STATUS_NOT_ENOUGH_DATA, "Premature end-of-file encountered.");
            }
        }
        VP8InitScanline(dec);   // Prepare for next scanline

        // Reconstruct, filter and emit the row.
        if (!VP8ProcessRow(dec, io))
        {
            ret_val = VP8SetError(dec, VP8_STATUS_USER_ABORT, "Output aborted.");
        }
    }
    if (dec->mt_method_ > 0)
    {
        if (!WebPGetWorkerInterface()->Sync(&dec->worker_))
            ret_val = 0;
    }
```

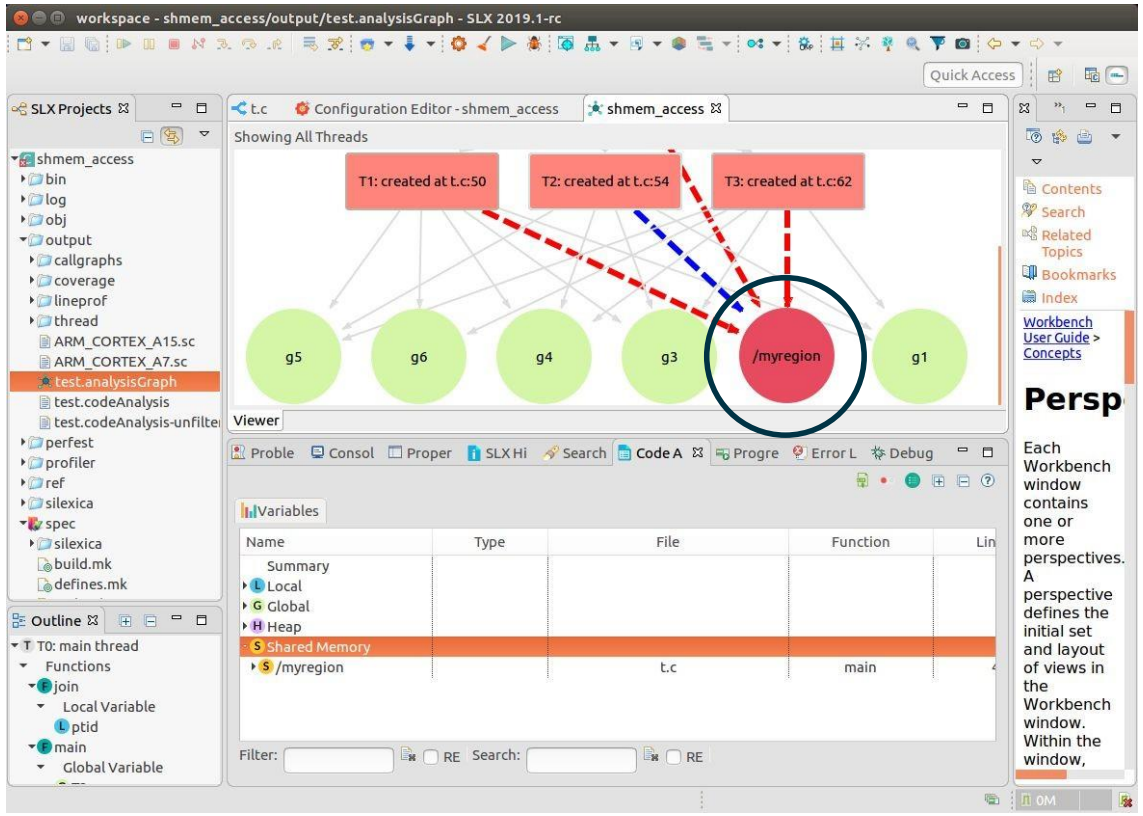
並列化ヒント

共有メモリ分析の可視化

CAG shows shared memory variables for processes and threads

- 共有メモリアクセスの統計情報を見ながら、分析対象の共有メモリと関連するスレッド・プロセスをコールグラフ上で確認
- ダブルクリック、もしくは右クリックで“Go to Source”を選択することで、対象のソースコードへジャンプ

⇒設計者は、分析対象を俯瞰しながら詳細分析が可能となり、最新のソフトウェアアーキテクチャの理解促進



共有メモリ保護解析
Shared Data Protection Analysis

メモリ保護のメカニズムの理解
- スレッド間・プロセス間の保護ミス
- 問題個所のソースコード行の特定
- どのmutexが変数を保護しているかの調査

Protection	Column Value	Description
Unprotected	*Unprotected*	All accesses happen without a mutex being locked.
Sometimes	*Sometimes*	Some accesses happen with a mutex being locked, others without.
Protected	*Mutex Source Location*	All accesses happen with a mutex being locked, the declaration location of the mutex is shown.

業界初

別々のプロセスとアプリケーション間の潜在的なデータ競合の検出

Variables

Name	Type	File	Function	Line	Size (Bytes)	Reads	Writes	Total Access	Accessed By	Protected
Summary										
Local										
Global										
Heap										
Shared Memory										
SHM_Sysram				200	31387904	936270 (6.63%)	238063 (2.58%)	1174333 (5.03%)	T0, T1, T2, T3, T4, T5, T6, T7	Sometimes
SHM_Parameters				84	474446	223820 (1.58%)	2 (0.00%)	223822 (0.96%)	T0, T5	Unprotected

共有メモリ保護解析

Shared Variable Analysis in the Memory Analysis Table

Shared variable report

Problem: Unprotected shared variable

Read/Write counts

Variable accessed by multiple threads

Variables

Name	Type	File	Function	Line	Reads	Writes	Total Access	Accessed By	Protected
Summary									
Local									
Global									
g1	int	t.c		10	55 (20.75%)	2 (1.90%)	57 (15.41%)	T1, T2, T3	Unprotected
g2	int	t.c		10	0 (0.00%)	1 (0.95%)	1 (0.27%)		
g3	int	t.c		10	3 (1.13%)	0 (0.00%)	3 (0.81%)	T1, T2, T3	Unprotected
g4	int	t.c		10	0 (0.00%)	3 (2.86%)	3 (0.81%)	T1, T2, T3	Unprotected
g5	int	t.c		10	3 (1.13%)	1 (0.95%)	4 (1.08%)	T1, T2, T3	Unprotected
g6	int	t.c		10	0 (0.00%)	2 (1.90%)	2 (0.54%)	T1, T3	Unprotected
T1	pthread_t	t.c		9	1 (0.38%)	1 (0.95%)	2 (0.54%)		
T2	pthread_t	t.c		9	1 (0.38%)	1 (0.95%)	2 (0.54%)		
T3	pthread_t	t.c		9	1 (0.38%)	1 (0.95%)	2 (0.54%)		
Heap									
malloc		t.c	task2	25	0 (0.00%)	2 (1.90%)	2 (0.54%)		Sometimes
malloc		t.c	main	42	1 (0.38%)	6 (5.71%)	7 (1.89%)	T1, T2, T3	Unprotected

Critical Bug?

SLXを使用するメリット:

→ すべてのスレッドで共有メモリ・変数がどのように使用されているか調査。不足している保護を検出。

→ 特定のユースケースにおける、共有メモリ・変数のアクセスタイプとアクセス頻度結果より、実装の振る舞いを理解。

→ 共有メモリ・変数に関連する性能のボトルネックを特定。

→ 実コードに基づいた動的解析による必要とされるメモリリソースの理解。