

GLIMPSES: Memory and program behavior estimation for SPEs

Jaswanth Sreeram, Ling Liu, Santosh Pande





Motivation

“Prototyping large codebases for porting to SPEs is challenging”

*“Need a way to quickly evaluate program behavior and its suitability for SPEs” –
Important for legacy code/reuse*



Motivation (contd)

- Porting large codebases to SPEs is challenging
 - Limited local store
 - High branch penalty
 - Geared towards vectorizable code
 - Code/data partitioning is not trivial
 - SPE-SPE, SPE-PPE interactions
- Provide programmer with tools to
 - Understand dynamic program behavior
 - Quickly construct candidate partitions for SPEs
 - Evaluate/Quantify partitions' suitability for SPEs



GLIMPSES: Tool Overview

- Dynamic Call Graphs
- Memory Requirements
 - Dynamic
 - Analytical
- Memory Access Patterns
 - Locality (spatial, temporal, neighbor affinity)
- Partitioning
 - Criteria based estimates
- Visual, interactive



Dynamic Call Chains

The screenshot shows a software window titled "Memory Estimator" with a "Data" tab. The main area displays a complex graph visualization of call chains, consisting of numerous nodes and edges. The graph is organized into several clusters, with a large central cluster and several smaller ones branching out. The nodes are small circles, and the edges are thin lines connecting them.

On the right side of the window, there is a control panel with several sections:

- NBodyForce**: Includes sliders for "Gravitational..." (set to 45), "Distance" (set to 0), and "BarnesHutTheta" (set to 90). The values -1.0 and 0.89 are shown next to the sliders.
- DragForce**: Includes a slider for "DragCoefficient" (set to 10) with a value of 0.00 shown.
- SpringForce**: Includes sliders for "SpringCoeffic..." (set to 9) with a value of 9.99, and "DefaultSpring..." (set to 25) with a value of 50.0.
- Connectivity Filter**: Includes a slider for "Distance" (set to 30) with a value of 30.

Below the control panel is a "Results Display Panel" containing the text "Hello World".

Annotations with arrows point to the "Graph Visualization Area" and the "Results Display Panel".



Call chains...contd

The screenshot displays a call chain diagram for the function `Decode_Bitstream.27`. The diagram shows a central node `Decode_Bitstream.27` (highlighted in red) with arrows pointing to various other functions. A prominent arrow points to `video_sequence.26`, which in turn points to `Initialize_Sequence.29`. Other functions shown include `read`, `Fill_Buffer`, `Flush_Buffer`, `Initialize_Buffer`, `Process_Options.32`, `toupper`, `atoi`, `open`, `next_start_code`, `Show_Bits`, `Headers.28`, `Flush_Buffer`, `Get`, and `Show_Bits`. The `main` function is also visible as a central hub for many of these calls.

On the right side, a panel provides details for the selected function `Decode_Bitstream`:

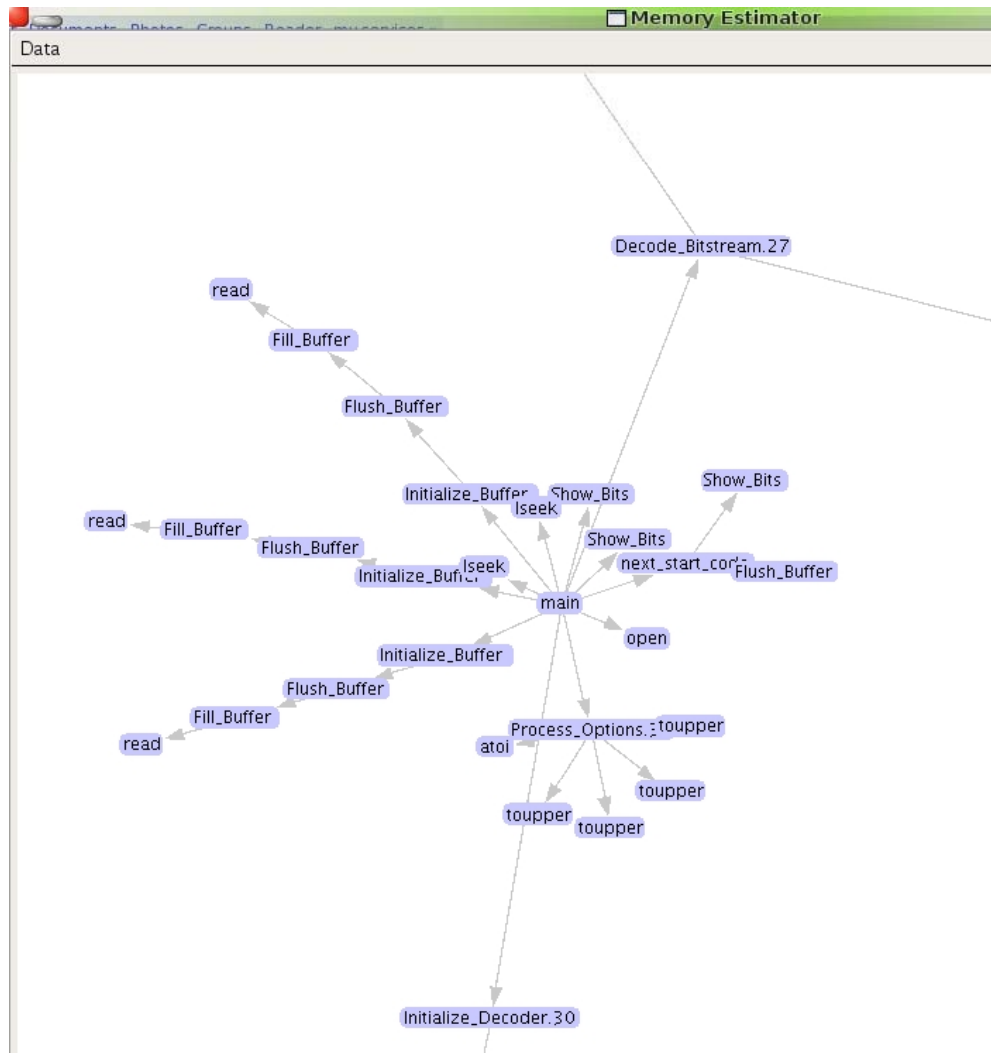
- Function Name: Decode_Bitstream
- Code Size (Bytes): 910
- Stack Size (Bytes): 4
- Branch Density: 0.2
- # of vectorizable loops: 0
- Dyn. Mem Alloc (Bytes): 337920
- WARNING: Local Store memory limit hit!

Below the function details, there are several control panels for different physics and simulation parameters:

- NBodyForce**: Gravitational... (45, -1.0), Distance (0, -1.0), BarnesHutTheta (90, 0.89)
- DragForce**: Drag Coefficient (27, .027)
- SpringForce**: Spring Coefficient (9, 9.99), Default Spring... (25, 50.0)
- Connectivity Filter**: Distance (30, 30)



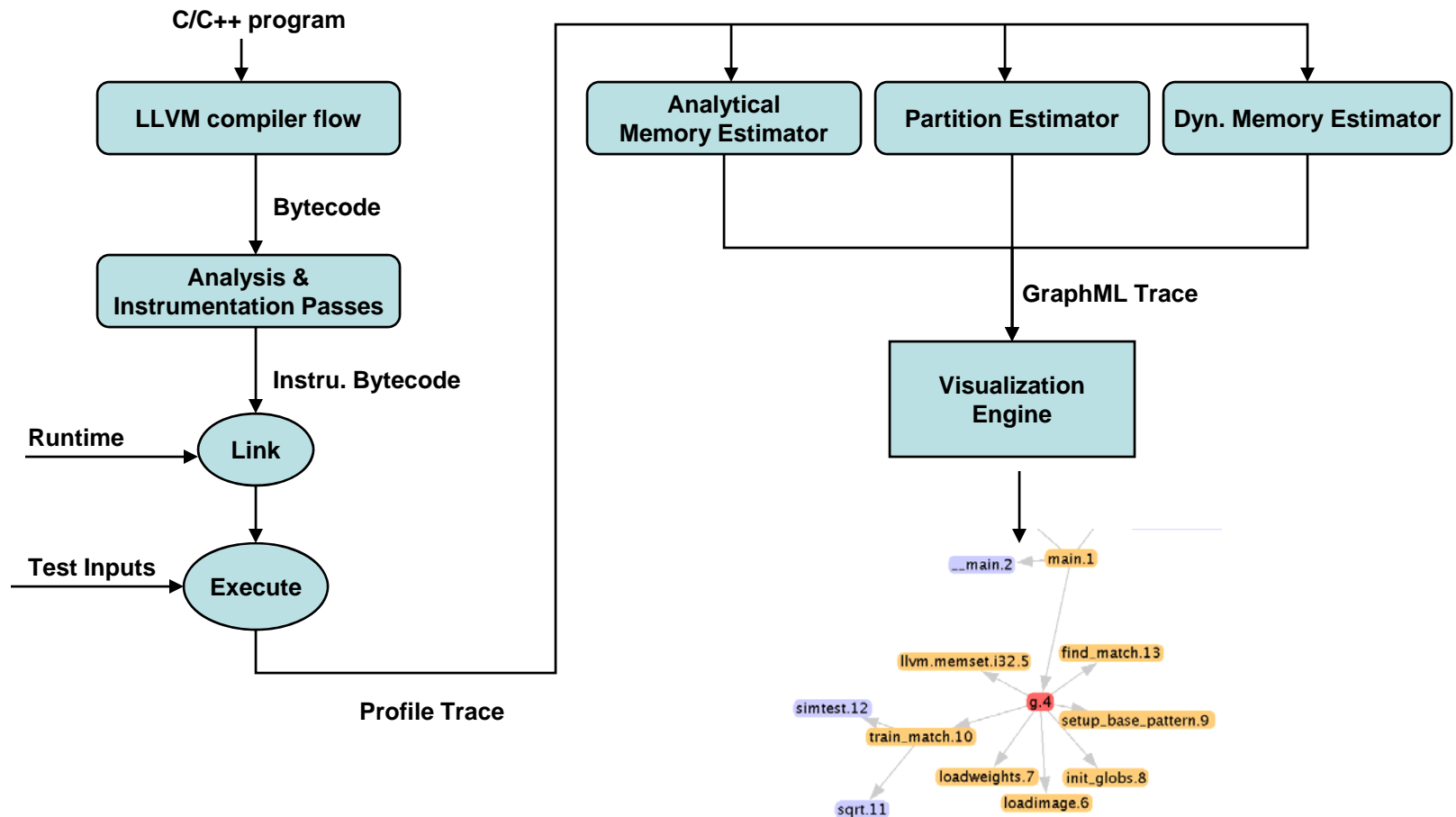
Mpeg-2 Decode



- Zoom view
- Shows dynamic call chains for a program run (in this case the program is mpeg2-decode)



GLIMPSES





Memory Behavior

- Estimate static and dyn. memory usage
 - Code, stack and heap (per function)
 - Usage < SPE LS limit ?
- Estimate function attributes
 - Branch density
 - Number of Auto-vectorizable loops
- Analytical estimation
 - Detect program objects affecting dynamic memory behavior
 - Show correlation between these program objects and memory usage.
 - Construct an arithmetic expression for amount of memory allocated, in terms of inputs or other program objects



Analytical Estimator : Mpeg2 example

Code segment

```
for (.....)
{
if (cc==0)
    size = Picture_Width*Picture_Height;
else
    size = Chroma_Width*Chroma_Height;

    if (!(backward_reference_frame[cc] =
(unsigned char *) malloc(size) ))
        Error(...);

    if (!(forward_reference_frame[cc] =
(unsigned char *) malloc(size) ))
        Error(...);
}
```

Result

```
__Malloc_size__1 = 1024

__Malloc_size__2 = 0 +
Coded_Picture_Width*Coded_Picture_Height

__Malloc_size__3 = 0 +
Coded_Picture_Width*Coded_Picture_Height

__Malloc_size__4 = 0 +
Coded_Picture_Width*Coded_Picture_Height

__Malloc_size__5 = 0 +
Chroma_Width*Chroma_Height

__Malloc_size__6 = 0 +
Chroma_Width*Chroma_Height

__Malloc_size__7 = 0 +
Chroma_Width*Chroma_Height

__Malloc_size__8 = 0 +
Chroma_Width*Chroma_Height
```

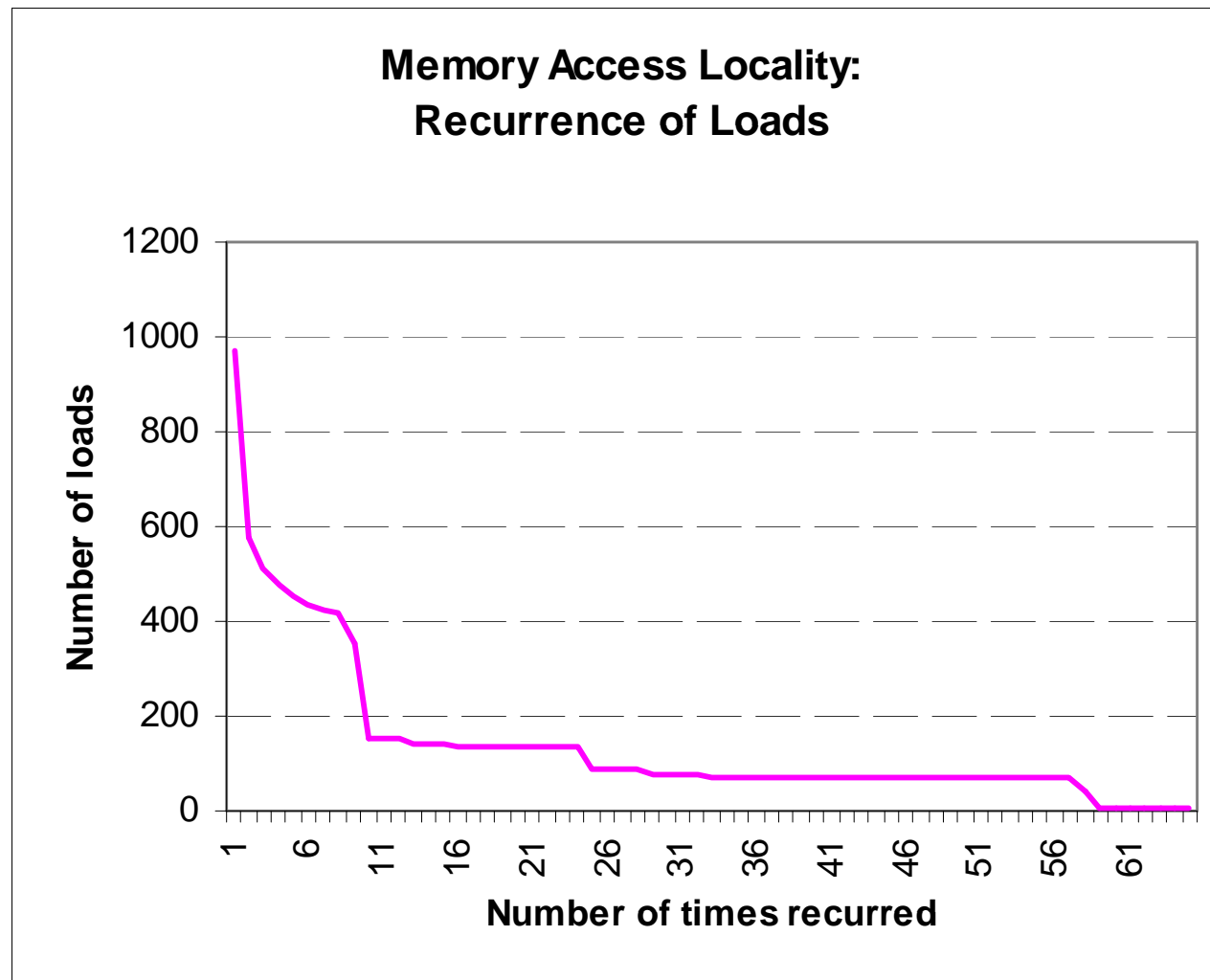


Memory Access Patterns

- Locality metrics for loads/stores
 - Spatial Locality
 - “Loads to different addresses in a spatial window”
 - Temporal Locality
 - “Loads to same address in a time window”
 - Neighbor Affinity
 - “Loads to addresses within a space and time window”

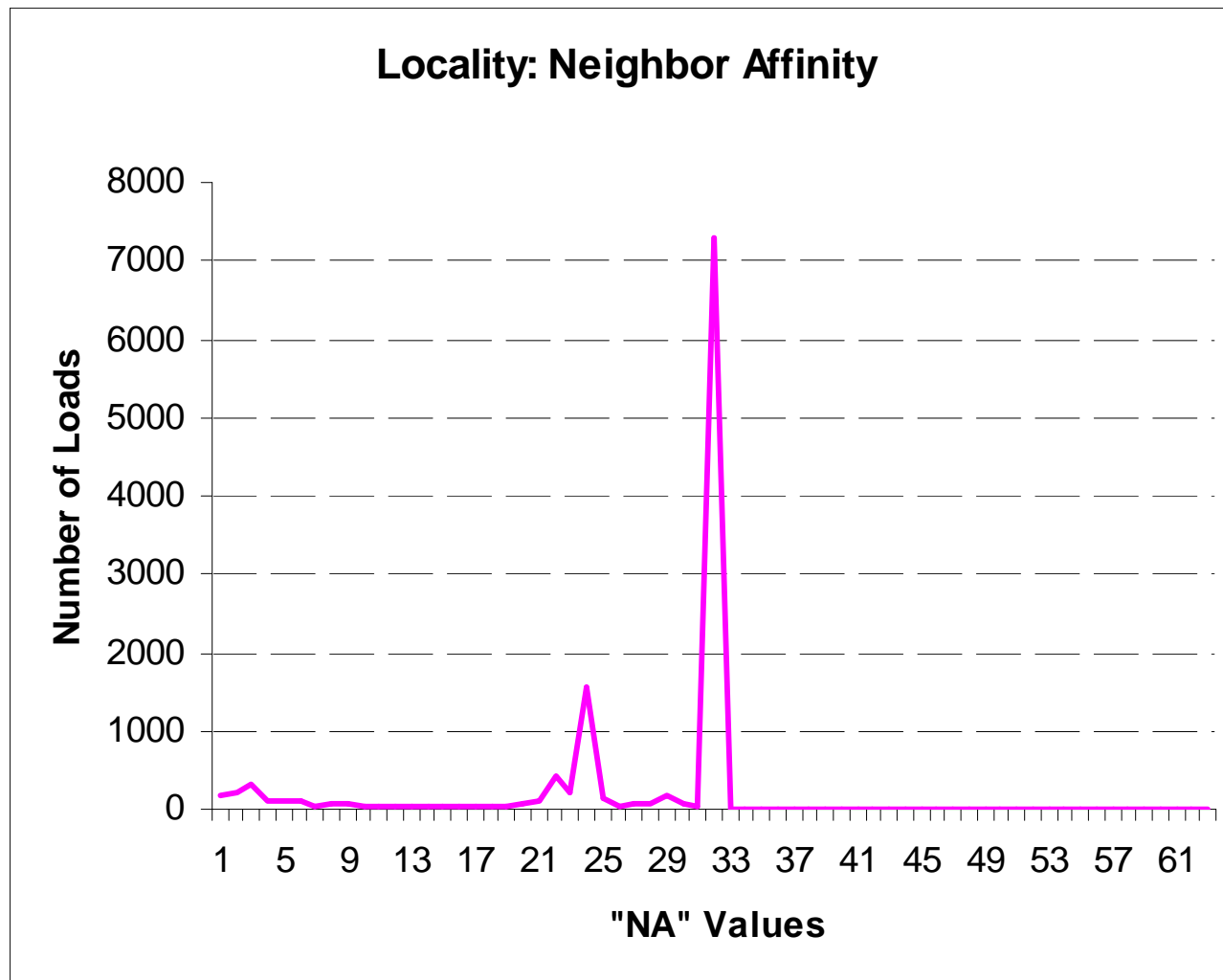


Locality Measures (mpeg2decode)





Locality measures: Affinity



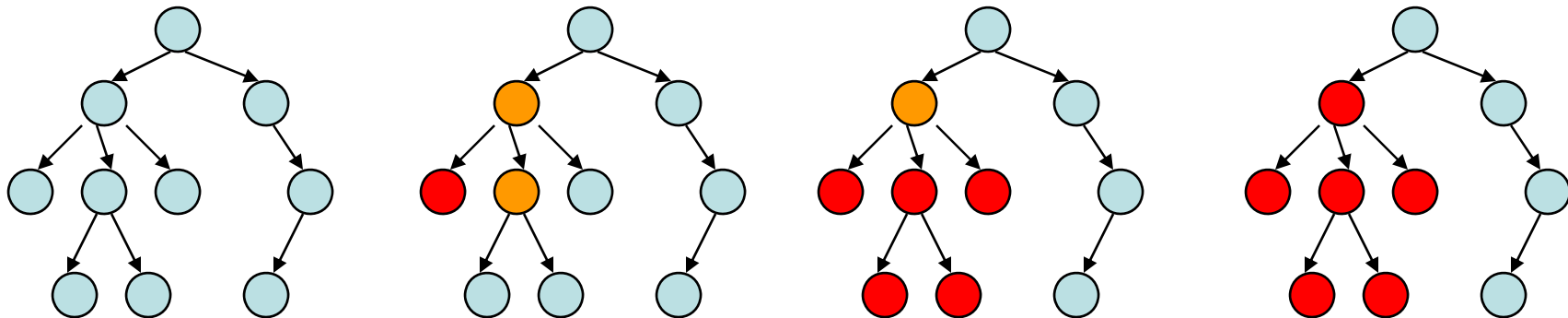


Program Partitions

- Provide programmer with possible partition candidates
 - Can be based on criteria:
 - Memory consumption
 - Memory reference behavior
 - Branch density
 - Auto-vectorizable loops
 - Aliasing
 - Combination (a “rank” metric)
 - Does not assume code/data overlays



Partitioning



- Start with earliest leaf node in dyn. call graph in a partition
- Try to add its parent to the partition
 - Try to add all of parent's children to the partition
 - If they can be added, try to add parent to partition.
- Try to add parent's parent to partition

- Estimates only: No code generation
- Programmer to take care of "cloning".
- Can produce interprocedural, context sensitive alias information.
 - Given two partitions, can they alias each other's data ?



Status

- Several features/improvements planned
 - Alias Analysis information for refining partition-set
 - Alias Analysis information for data pinning/prefetching opportunities.
 - Leverage **DataStructureAnalyses** for smart memory allocation on SPUs
- Tested on
 - Workloads from SPECINT
 - Workloads from mediabench
 - ODE (Open Dynamics Engine)
- Beta version to be released shortly.



The End

Email contact:

jaswanth@cc.gatech.edu