HIGH-PERFORMANCE SECURITY APPLICATIONS: CHALLENGES, ALGORITHMS, AND FRAMEWORKS

Xiaodong Yu (final year Ph.D. student)

Danfeng (Daphne) Yao (Ph.D. advisor)

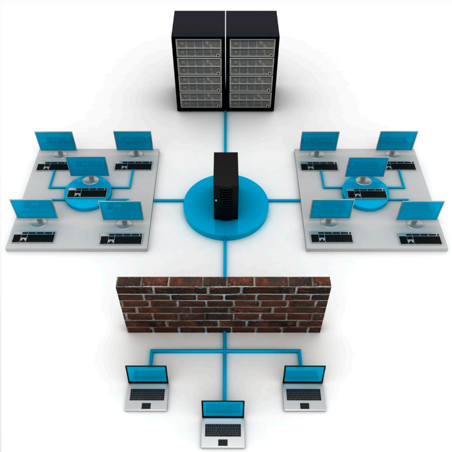
Dept. of Computer Science, Virginia Tech

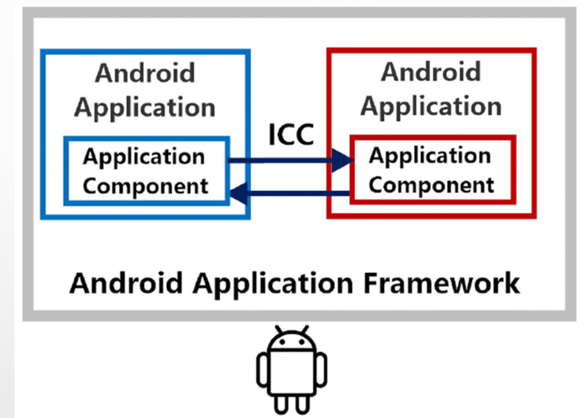
Introduction

Cyber-security community

scalability and timeliness requirements in real-life scenarios

Firewall&NIDS	Data leaks detection	Android ICC analysis





Firewall&NIDS: Line rate; Offloading and buffering

Data leaks detection: Real time; Hours level delay

Android ICC analysis: Nightly; 6340 hours [AsiaCCS’17]

Introduction

Research trend in cyber-security community

Three years ('15-'17) stats of top-tier security conference publications

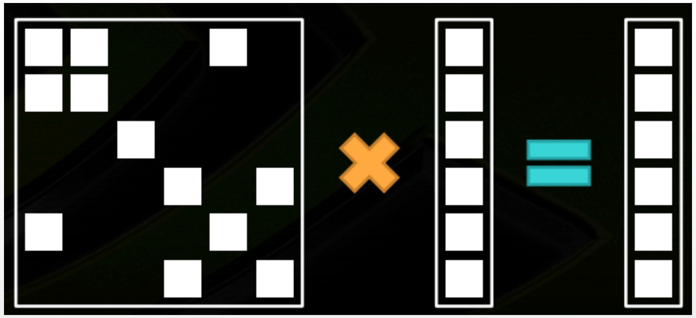
	Total # of papers	# of papers emphasize on the algorithm efficiency and scalability	# of papers focus on implementations on real HPC
CCS	416	49 (11.8%)	19 (2.2%)
IEEE S&P	180	12 (6.7%)	10 (5.6%)
USENIX Security	224	12 (5.4%)	9 (4.0%)
NDSS	178	11 (6.2%)	4 (2.2%)

Introduction

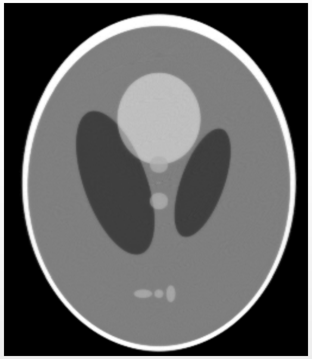
HPC community

– Require applicaition-specific tuning in addition to generic algorithm optimization to achieve optimal performance

Sparse matrix-vector multiplication (SpMV)



CT image reconstruction



cuSPARSE (3x vs. CPU)

w/ app.-specific opt. (21x vs. CPU)

cuSPARSE (up to 15x vs. CPU)

Mainly for bioinformatics, data mining, climate predicting etc.

Barely for cyber security applications.

Introduction

Research trend in HPC community

Three years ('15-'17) stats of top-tier HPC conference publications

	Total # of papers	# of papers focus on generic algorithms impl.&opt. on HPC	# of papers focus on applications (security apps) impl.&opt. on HPC
SC	221	95 (43.0%)	29(0) (13.1%(0%))
ICS	111	34 (30.6%)	6(1) (5.4%(0.9%))
HPDC	77	25 (32.5%)	8(1) (10.4%(1.3%))
IPDPS	339	97 (28.6%)	37(3) (10.9%(0.9%))

Thesis Goal

Narrow the gap between cybersecurity and HPC communities in three dimensions

1. Identify the challenges
2. Design or refactor the algorithms
3. Provide the implementation and optimization frameworks

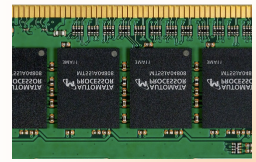
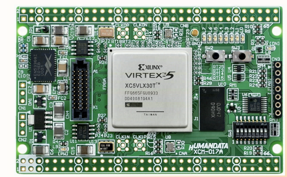
Targeted Algorithms

Two algorithms that are broadly adopted in a wide range of cybersecurity applications

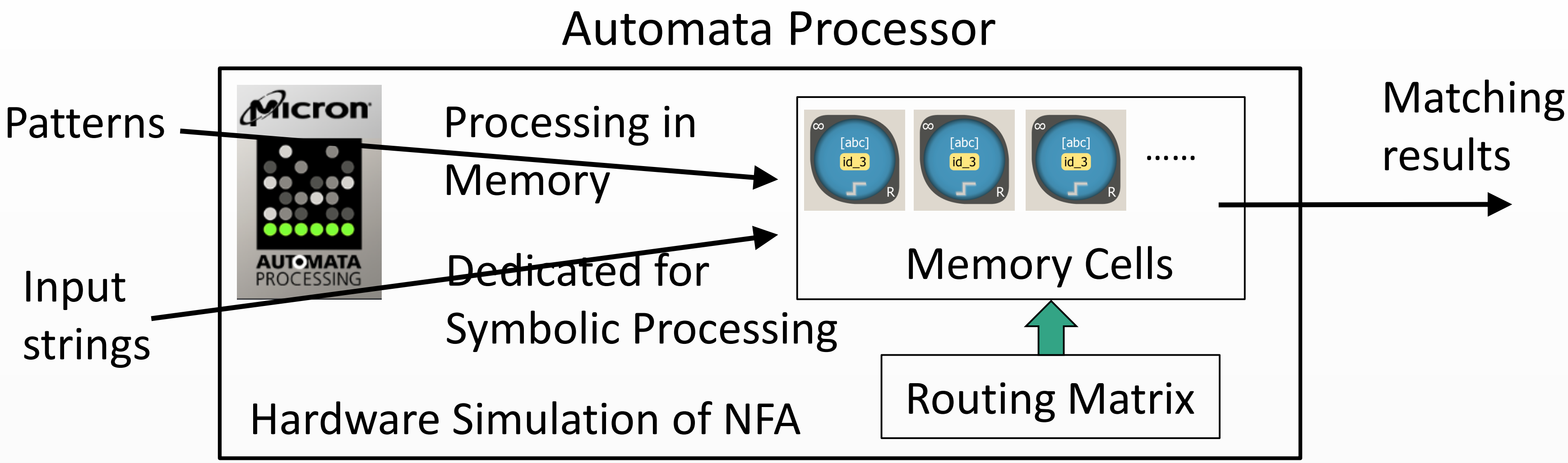
- **Automata Processing**
  1. Intrusion Detection/Prevention System (IDS/IPS)
  2. Anomaly detection
  3. Packet filtering/Deep Packet Inspection (DPI)

- **Worklist algorithm**
  1. Taint analysis
  2. Point-to analysis
  3. Control-flow integrity

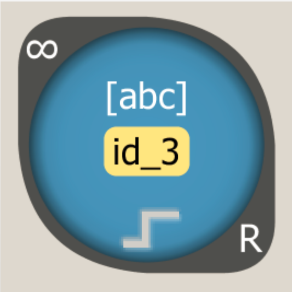
Targeted Platforms

<table>
<tr><th></th><th>GPU</th><th>Micron’s AP</th><th>FPGA</th></tr>
<tr><td></td><td></td><td></td><td></td></tr>
<tr><td>ENCODING /states</td><td colspan="2">memory</td><td></td></tr>
<tr><td>ENCODING/transitions</td><td></td><td colspan="2">logic</td></tr>
<tr><td rowspan="2">PROGRAMMING</td><td colspan="2">memory configuration</td><td></td></tr>
<tr><td></td><td colspan="2">compilation, place&route</td></tr>
<tr><td>TRAVERSAL BEHAVIOR</td><td>strongly dataset dependent
(active state set matters)</td><td colspan="2">weakly dataset dependent
(input character/clock cycle + output processing)</td></tr>
</table>

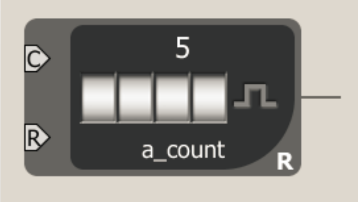
Introduction to Automata Processor (AP)



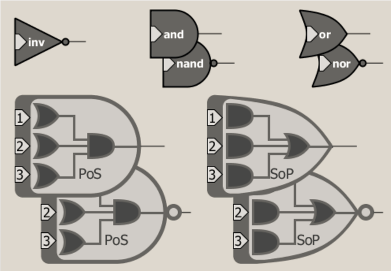
Three Programmable Resources



State Transition Element (STE)
49152 per chip



Counter Element
768 per chip



Boolean Element
2304 per chip

Contribution Summary

Algorithms

1. Automatically Building Augmented-FA to Solve State Blow-up in matching core of DPI [*IEEE Journal on Selected Areas in Communications (IF: 7.17) – Journal first paper*]
2. O^3FA: A Scalable Finite Automata-based Pattern-Matching Engine for Out-of-Order DPI [*ACM/IEEE ANCS'16*]

Frameworks

1. GPU Acceleration of Regular Expression Matching for Large Datasets [*ACM CF'13*]
2. Robotomata: A Framework for Approximate Pattern Matching of Big Data on an Automata Processor [*IEEE BigData'17*]
3. Demystifying automata processing: GPUs, FPGAs or Micron's AP? [*ACM ICS'17*]
4. GPU-assisted Android Program Analysis Framework (ongoing work)

Contribution Summary

Algorithms

1. Automatically Building Augmented-FA to Solve State Blow-up in matching core of DPI [*IEEE Journal on Selected Areas in Communications (IF: 7.17) – Journal first paper*]
2. O^3FA: A Scalable Finite Automata-based Pattern-Matching Engine for Out-of-Order DPI [*ACM/IEEE ANCS'16*]

Frameworks

1. GPU Acceleration of Regular Expression Matching for Large Datasets [*ACM CF'13*]
2. Robotomata: A Framework for Approximate Pattern Matching of Big Data on an Automata Processor [*IEEE BigData'17*]
3. Demystifying automata processing: GPUs, FPGAs or Micron's AP? [*ACM ICS'17*]
4. GPU-assisted Android Program Analysis Framework (ongoing work)

O^3FA: A Scalable Finite Automata-based Pattern-Matching Engine for Out-of-Order DPI

Xiaodong Yu 11/13/2018
SC'18 Doctoral Showcase

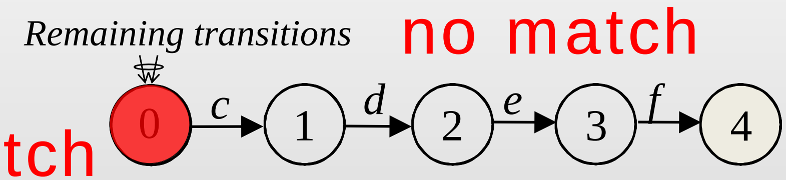
Problem Definition

- Input stream: MB or even GB level
- Single packet payload is only 64KB
  - → multiple substreams→ carried by different packets

- **Out-of-Order Packets:** packets arrive to the destination in a random order
  - → match escape from the NIDS due to cross-packet matching

input stream: *abcdefgh* malicious pattern: *cdef*

no match



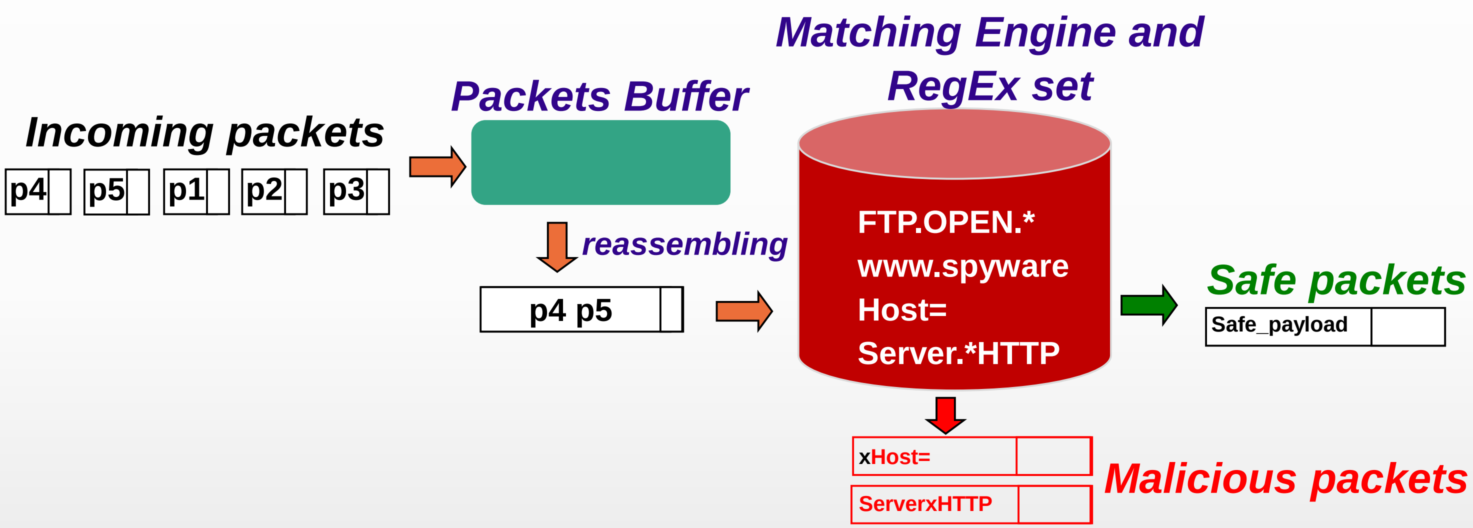
match *cdef*

no match

abcdefgh → *efgh* ⇒ *abcd*

Traditional Approach

Buffering and reassembling: Industrial standard solution for the Out-of-Order Deep Packet Inspection (DPI)



Drawback:

1. Very resource-intensive
2. Attackers can exhaust the packet buffer capacity

O^{3}FA Design

Input stream: *aabcdefg* malicious pattern: *bcde*

One match: *cabcdeab*

Two packets: *P1=aabc* *P2=defg*

Arrived in reversed order: *P2* ⇨ *P1*

Segments: *P2=defg* *P1=aabc*

suffix of *bcde* prefix of *bcde*

When *P2* arrives, detect and record only *de*

When *P1* arrivew, retrieve *de* and concatenate it to *P1*

Then we can reconstruct the match *aabcde*

O^3FA Design

Advanced challenge: complex sub-patterns in RegEx

- the most general unbounded repetition: dot-star terms (.*)

Difficulties to handle dot-star terms:

How to represent and detect prefix/suffix of malicious patterns that contain dot-star terms?

construct FA for prefix/suffix set that contain dot-star terms

How to retrieve the segment through recorded information that matched prefix/suffix with dot-star terms?

Restore the shortest segment (may not be the same as original segment) that can match the same prefix/suffix

O³FA Design

Observation:

Input stream: *aabafgcdef* malicious pattern: *.*b.*cde*

One match: *aabafgcdef*

Two packets: *P1*=*aabaf* *P2*=*gcdef*

Arrived in reversed order: *P2* ⇨ *P1*

Segments: *P2*=*gcdef* *P1*=*aabaf*

match suffix *.*cde* Match prefix of *.*b.**

When *P2* arrived, suffix *.*cde*

shortest segment *cde*

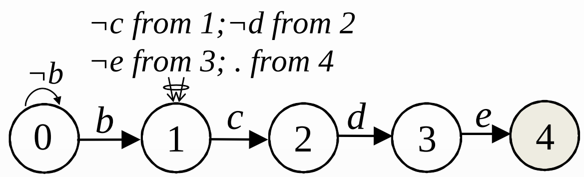
P1 → *P1'*=*aabafcde*

segment *gcde* and *cde* are equivalent

O^{3}FA Design

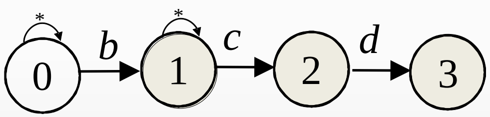
- O^{3}FA (Out-of-Order Finite Automata)

Regular DFA

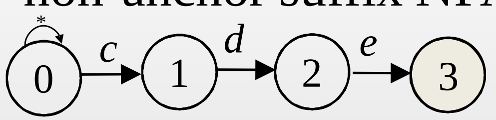


Supporting-FAs

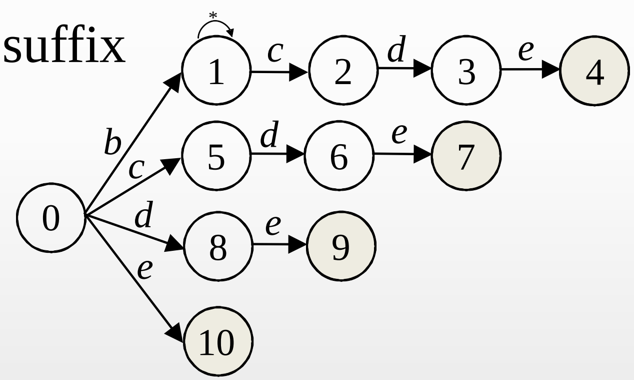
prefix NFA



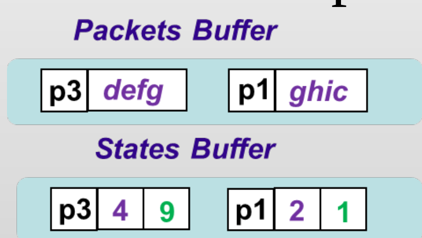
non-anchor suffix NFA



anchor suffix NFA



- States buffer instead of packets buffer



8 Bytes

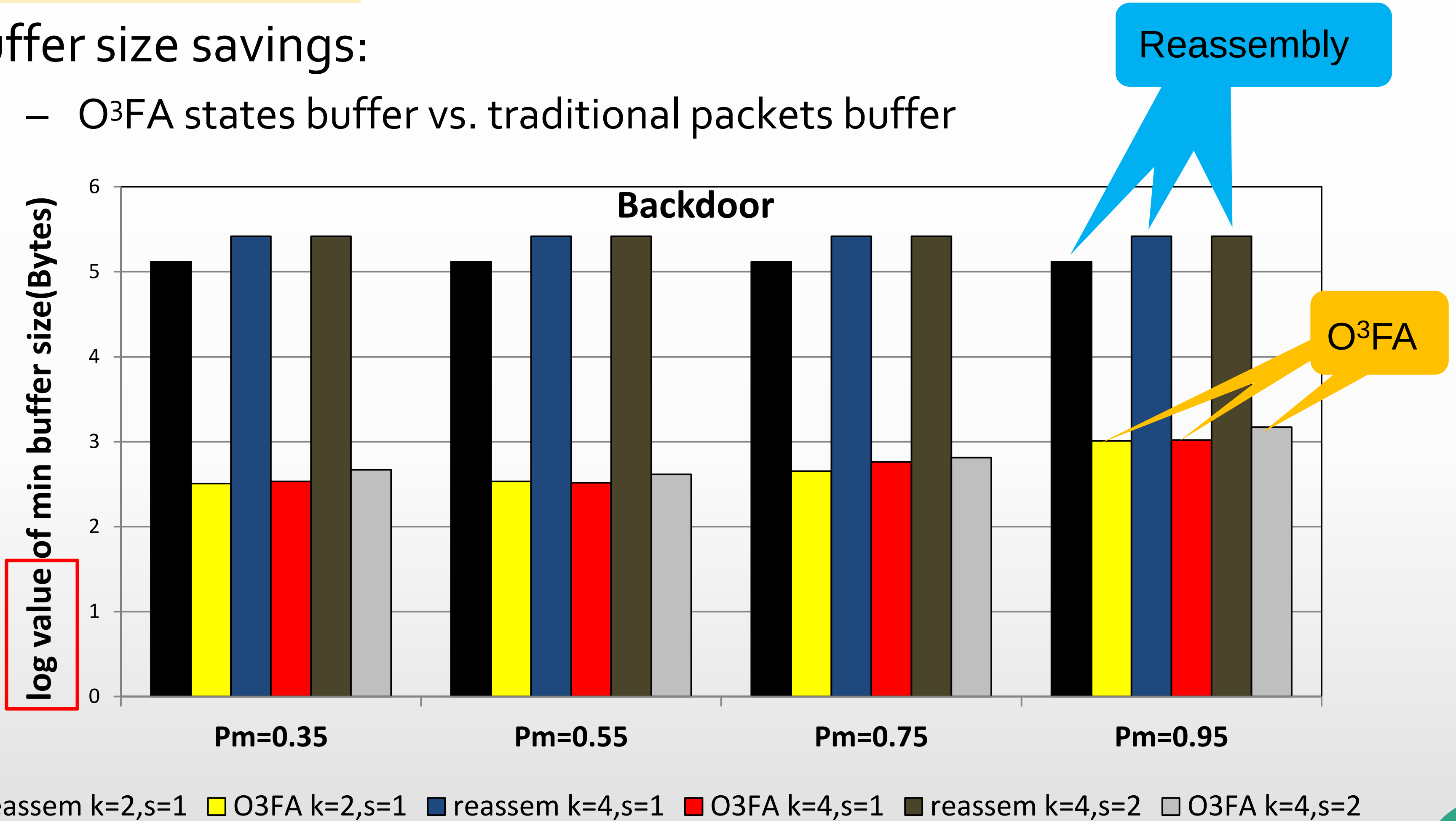
4 Bytes

VirginiaTech Invent the Future

Evaluation

Buffer size savings:

– O3FA states buffer vs. traditional packets buffer



State buffer requires 20x-4000x less size than packet buffer

Robotomata: A Framework for Approximate Pattern Matching of Big Data on an Automata Processor

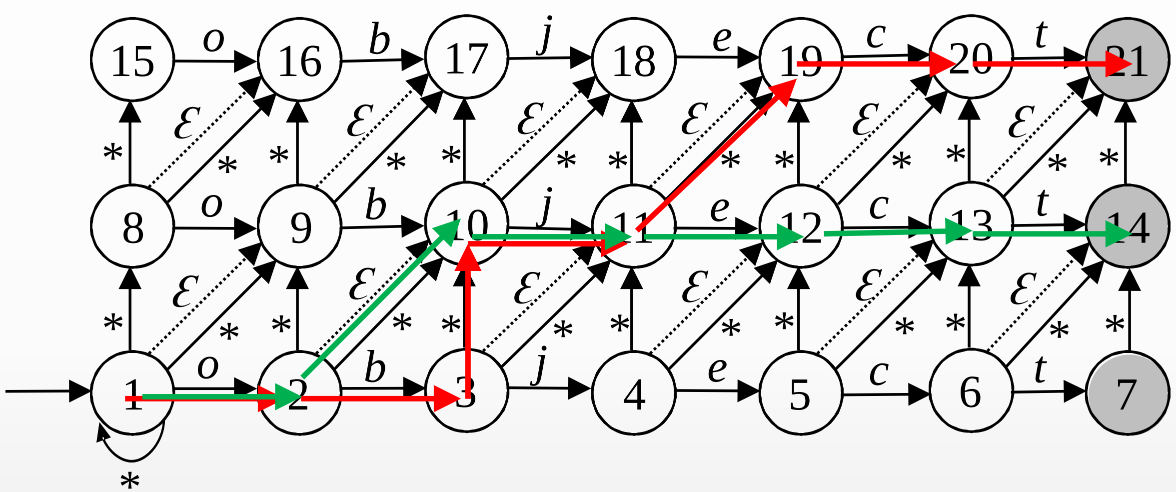
Xiaodong Yu 11/13/2018
SC'18 Doctoral Showcase

VirginiaTech
Invent the Future

Automata-based APM: Levenshtein Automaton

Target pattern: *object*

… which allows a Levenshtein distance of up to 2



ε allows automaton to change its state spontaneously, i.e. without consuming an input symbol

* allows automaton to be triggered by any input character

Input string: *oject* Output: detected (with one error → D)

Input string: *obijfct* Output: detected (with two errors → I, S)

VirginiaTech Invent the Future

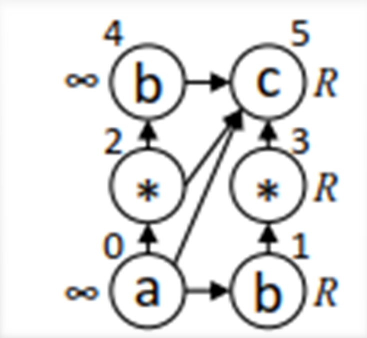
Challenges of APM on AP

Programmability

Mathematical format

ANML

- ANML: Automata Network Markup Lang
  - Requires expertise with both automata t
  - Example

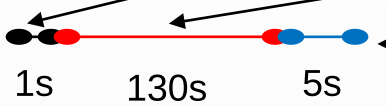


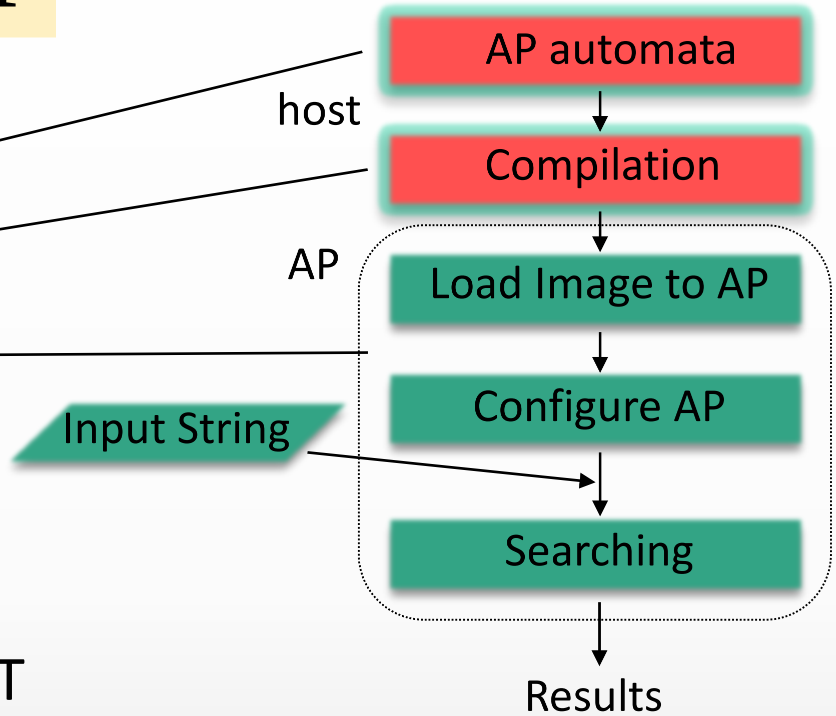

```
ap_anml_t anml = 0;
ap_anml_network_t anmlNet;
struct ap_anml_element element;
ap_anml_element_ref_t element[5];
ap_automaton_t a;
ap_element_map_t elementMap;

// Create the ANML object
anml = AP_CreateAnml();

// Create the automata network in the ANML object
AP_CreateAutomataNetwork(anml, &anmlNet, "an1");
// Create the elements that match "a" and start the search
element.res_type = RT_STE;
element.start = START_OF_DATA;
element.symbols = "a";
element.match = 0;
AP_AddAnmlElement(anmlNet, &element[0], &element);
// Create the elements that match "b" report the match
element.res_type = RT_STE;
element.start = NO_START;
element.symbols = "b";
element.match = 1;
AP_AddAnmlElement(anmlNet, &element[1], &element);
// The rest four STEs are created in the same manner
// with different symbols attribute
 ......
// Connect the STEs together to search "abc"
// and allow one Levenshtein distance
// Match
AP_AddAnmlEdge(anmlNet, element[0], element[1], 0);
AP_AddAnmlEdge(anmlNet, element[4], element[5], 0);
// Insertion
AP_AddAnmlEdge(anmlNet, element[0], element[2], 0);
AP_AddAnmlEdge(anmlNet, element[2], element[4], 0);
AP_AddAnmlEdge(anmlNet, element[1], element[3], 0);
AP_AddAnmlEdge(anmlNet, element[3], element[5], 0);
// Substitution
AP_AddAnmlEdge(anmlNet, element[2], element[5], 0);
// Deletion
AP_AddAnmlEdge(anmlNet, element[0], element[5], 0);
// Compile the ANML object to create the automaton
AP_CompileAnml(anml, &a, &elementMap, 0, 0, options, 0);
return 0;
```


VirginiaTech
Invent the Future

Challenges of APM on AP

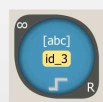
End-to-End Performance

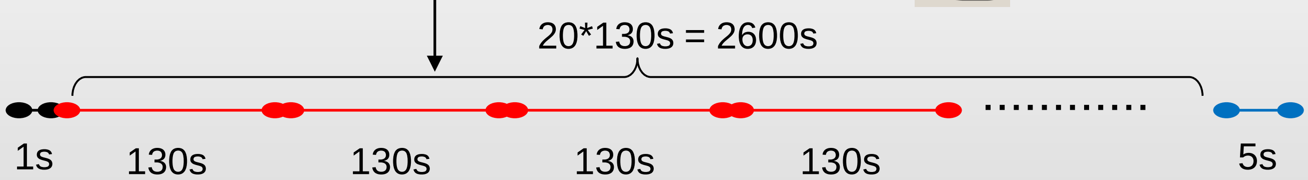




Example: *alua* dataset in BLAST

460K automata states vs. 50K STEs





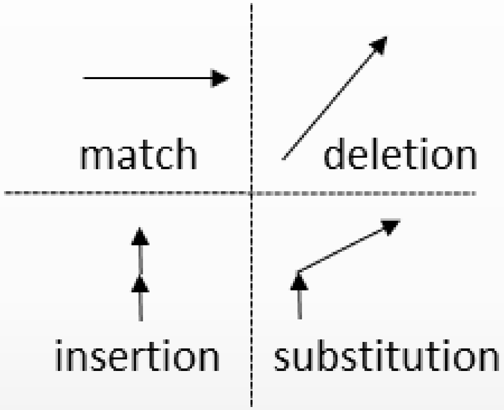
VirginiaTech
Invent the Future

Robotomata (Ro'-bo-tom'-ata)

Paradigm-based approach

– Four (4) paradigms: 3 error types (S,I,D) + 1 match type (M)

Distance	Paradigm
Levenshtein	M, S, I, D
Hamming	M, S
Episode	M, I
Longest Common Subsequence	M, I, D



- ➢ End-users only need to provide {***error types, pattern length, error number***}
- ➢ *Robotomata* generates the AP automata via a hierarchical construction

Robotomata: Hierarchical Construction

Paradigm set

Error types

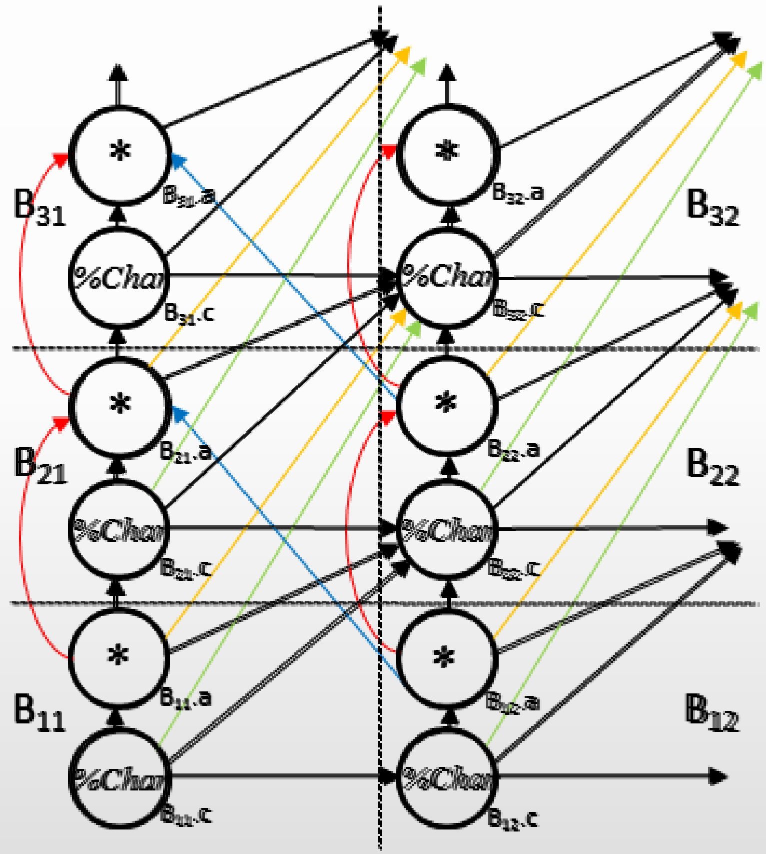
Building block

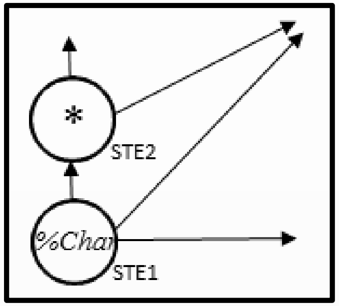
Length, Error#

Block matrix

Inter-block connection

AP Automata





VirginiaTech Invent the Future

Robotomata: Hierarchical Construction

In back end and transparent to end users!!!

{error types, pattern length, error#}

ROBOTOMATA

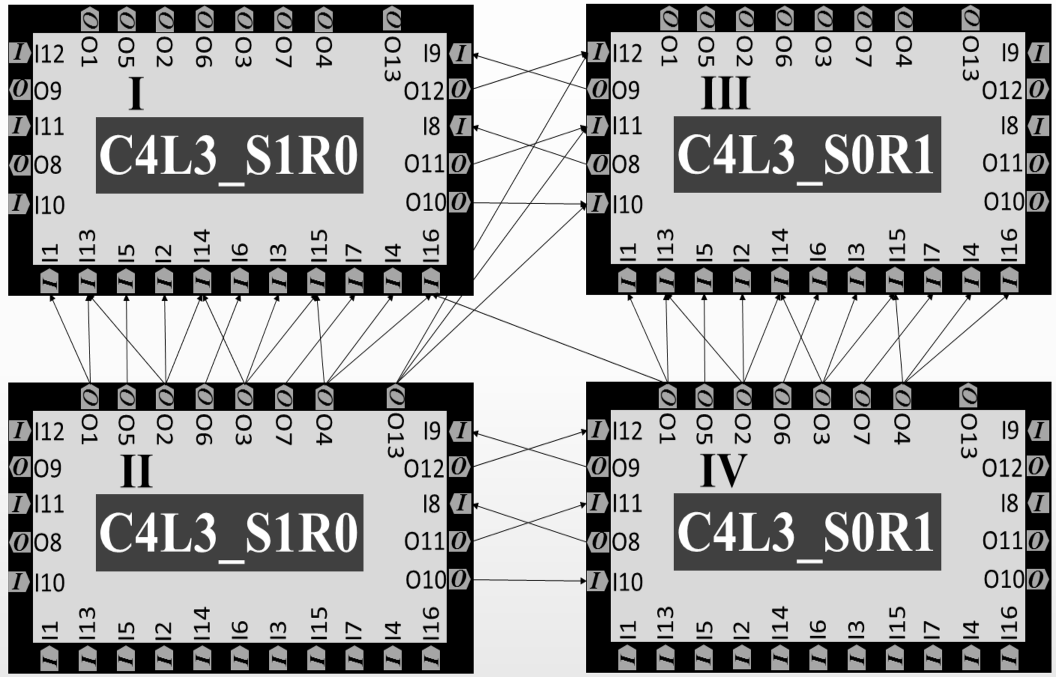
vs.

ANML API

```
ap_anml_t anml = 0;
ap_anml_network_t anmlNet;
struct ap_anml_element element;
ap_anml_element_ref_t element[5];
ap_automaton_t a;
ap_element_map_t elementMap;

// Create the ANML object
anml = AP_CreateAnml();

// Create the automata network in the ANML object
AP_CreateAutomataNetwork(anml, &anmlNet, "an1");
// Create the elements that match "a" and start the search
element.res_type = RT_STE;
element.start = START_OF_DATA;
element.symbols = "a";
element.match = 0;
AP_AddAnmlElement(anmlNet, &element[0], &element);
// Create the elements that match "b" report the match
element.res_type = RT_STE;
element.start = NO_START;
element.symbols = "b";
element.match = 1;
AP_AddAnmlElement(anmlNet, &element[1], &element);
// The rest four STEs are created in the same manner
// with different symbols attribute
 ......
// Connect the STEs together to search "abc"
// and allow one Levenshtein distance
// Match
AP_AddAnmlEdge(anmlNet, element[0], element[1], 0);
AP_AddAnmlEdge(anmlNet, element[4], element[5], 0);
// Insertion
AP_AddAnmlEdge(anmlNet, element[0], element[2], 0);
AP_AddAnmlEdge(anmlNet, element[2], element[4], 0);
AP_AddAnmlEdge(anmlNet, element[1], element[3], 0);
AP_AddAnmlEdge(anmlNet, element[3], element[5], 0);
// Substitution
AP_AddAnmlEdge(anmlNet, element[2], element[5], 0);
// Deletion
AP_AddAnmlEdge(anmlNet, element[0], element[5], 0);
// Compile the ANML object to create the automaton
AP_CompileAnml(anml, &a, &elementMap, 0, 0, options, 0);
return 0;
```


VirginiaTech
Invent the Future

Single BIG Automaton vs. Robotomata Cascadable Macro

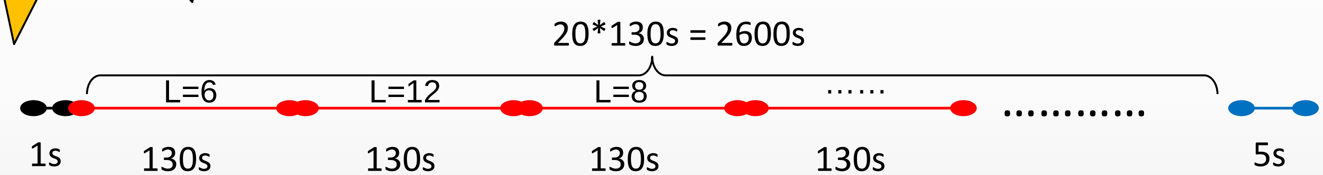


Robotomata: Cascadable Macros

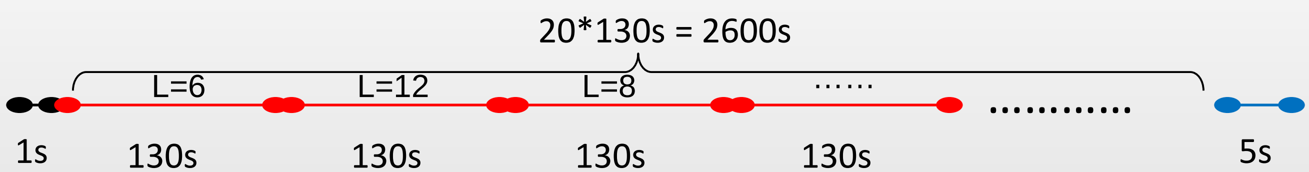
In back end and transparent to end users!!!

Enable cascadable macros

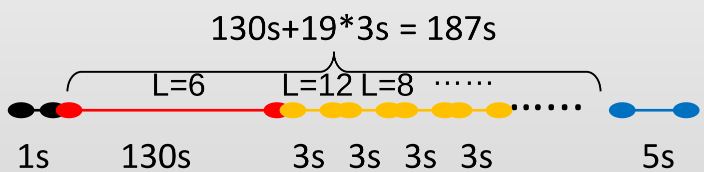
AP automata $\overset{\text{structures}}{\neq}$ macros



Classical macros



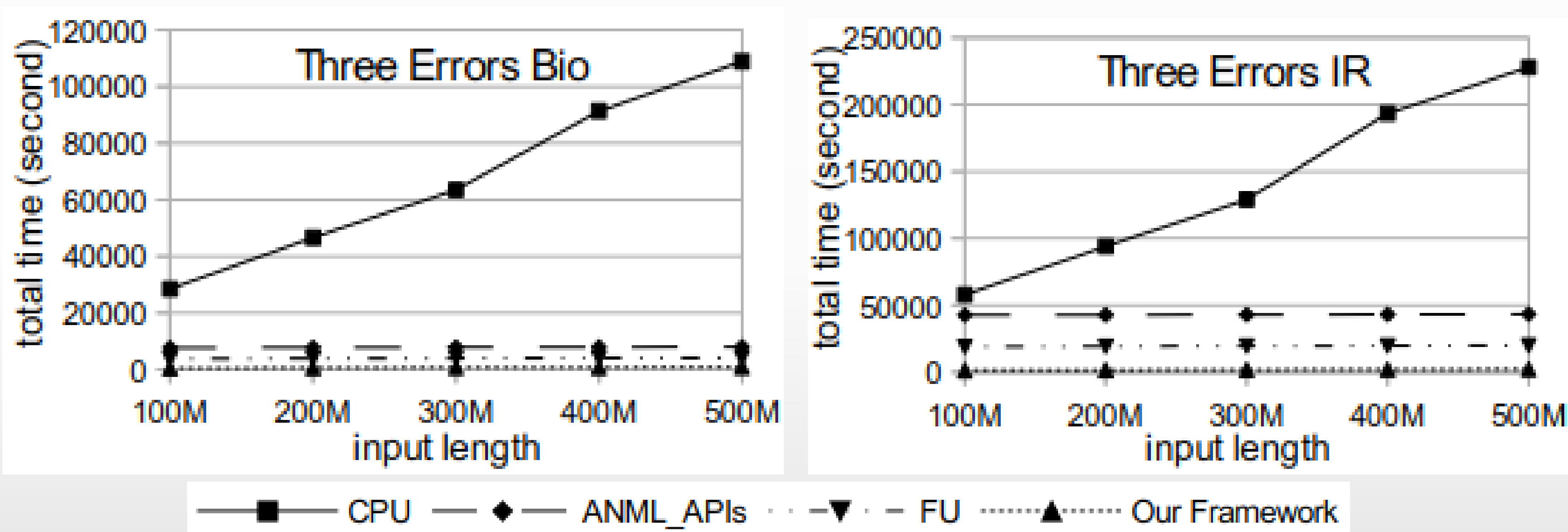
Cascadable macros



Evaluation

Overall performance comparison

- Bio: BLAST igseqprot dataset, 85k patterns
- IR: NHTSA Complaints dataset, 100k patterns



ROBOTOMATA can achieve up to 461x speedup over its CPU counterpart and 33.1x and 14.8x speedups over the ANML and FU-based Automata Processor (AP) versions, respectively

Side Projects

1. **GPU-based Computed-Tomography Image Reconstruction**
   - [*IEEE/ACM CCGrid'16*] [*ACM CF'17*] [*JSPS'18(journal)*]
   - Computational core is SpMV
   - Symmetry-based (app.-aware) compression in addition to CSR format
   - A uniform CUDA kernel handling both SpMV and SpMV_T
2. **A Data Layout-aware Auto-tuning Framework for Faster Convolutions on ROCm Platform**
   - Work done during the internship at AMD
   - Paper in submission
3. **Systematic Study of the Relevance between Cache Configuration and Side-channel Attack** (ongoing work)

Thank you!

Questions?