CLUSTERING STATIC ANALYSIS DEFECT REPORTS TO REDUCE MAINTENANCE COSTS

Zachary P. Fry and Westley Weimer

University of Virginia

Static Analysis-based Bug Finders

- Use known-faulty semantic patterns to find suspected bugs *statically*
  - Generally with minimal human intervention
- Valgrind, Fortify, SLAM, ConQAT, CodeSonar, PMD, Findbugs, Coverity SAVE, etc.
- Influential in both academia and industry
  - *Many* academic tools spanning various languages
  - Coverity boasts over 300 employees and over 1,100 customers, with extremely high growth

Static Analysis-based Bug Finders

- Produce many defect reports in practice

Program	KLOC	Reports
Eclipse	3,618	**4,345**
Linux (sound)	420	**869**
Blender	996	**827**
GDB	1,689	**827**
MPlayer	845	**500**

- Difficult to adapt to particular styles or idioms
- Regardless of true or false positives, groups of defect reports exhibit similarity in practice

Structurally Similar Defects

- Some defect reports are obviously similar or different
- Some are not:

```
printk(KERN_DEBUG "Receive CCP
  frame from peer slot(%d)",
  lp->ppp_slot);
if (lp->ppp_slot < 0 ||
    lp->ppp_slot > ISDN_MAX) {
  printk(KERN_ERR "%s:
    lp->ppp_slot (%d) out of
    range", _FUNCTION_,
    lp->ppp_slot);
  return;
}
is = ippp_table[lp->ppp_slot];
isdn_ppp_frame_log('ccp-rcv',
  skb->data, skb->len, 32,
```

```
if (!lp->master)
  qdisc_reset(lp->netdev->
    dev.qdisc);
lp->dialstate = 0;
dev->st_netdev[isdn_dc2minor(
            lp->isdn_device
            lp->isdn_channel)
              ] = NULL;
isdn_free_channel(
    lp->isdn_device,
    lp->isdn_channel,
    ISDN_USAGE_NET);
lp->flags &=
  ISDN_NET_CONNECTED;
```

```
sidx = isdn_dc2minor(di, 1);
#ifdef ISDN_DEBUG_NET_ICALL
 printk(KERN_DEBUG “n_fi:ch=0\n”);
#endif

if (USG_NONE(dev->usage[sidx])){
 if (dev->usage[sidx] &
       ISDN_USAGE_EXCLUSIVE) {
   printk(KERN_DEBUG “n_fi: 2nd
    channel is down and bound\n”);
   if ((lp->pre_device == di) &&
        (lp->pre_channel == 1)) {
```

Determining Defect Report Similarity

- Some defect reports are obviously similar or different
- Some are not:

```
printk(KERN_DEBUG "Receive CCP
  frame from peer slot(%d)",
  lp->ppp_slot);
if (lp->ppp_slot < 0 ||
    lp->ppp_slot > ISDN_MAX) {
  printk(KERN_ERR "%s:
    lp->ppp_slot (%d) out of
    range", _FUNCTION_,
    lp->ppp_slot);
  return;
}
is = ippp_table[lp->ppp_slot];
isdn_ppp_frame_log('ccp-rcv',
  skb->data, skb->len, 32,
```

```
if (!lp->master)
  qdisc_reset(lp->netdev->
    dev.qdisc);
lp->dialstate = 0;
dev->st_netdev[isdn_dc2minor(
            lp->isdn_device
            lp->isdn_channel)
            ] = NULL;
isdn_free_channel(
    lp->isdn_device,
    lp->isdn_channel,
    ISDN_USAGE_NET);
lp->flags &=
  ISDN_NET_CONNECTED;
```

```
sidx = isdn_dc2minor(di, 1);
#ifdef ISDN_DEBUG_NET_ICALL
 printk(KERN_DEBUG "n_fi:ch=0\n");
#endif

if (USG_NONE(dev->usage[sidx])){
 if (dev->usage[sidx] &
       ISDN_USAGE_EXCLUSIVE) {
   printk(KERN_DEBUG "n_fi: 2nd
    channel is down and bound\n");
   if ((lp->pre_device == di) &&
       (lp->pre_channel == 1)) {
```

Determining Defect Report Similarity

- Some defect reports are obviously similar or different
- Some are not:

```
printk(KERN_DEBUG "Receive CCP
  frame from peer slot(%d)",
  lp->ppp_slot);
if (lp->ppp_slot < 0 ||
    lp->ppp_slot > ISDN_MAX) {
  printk(KERN_ERR "%s:
    lp->ppp_slot (%d) out of
    range", _FUNCTION_,
    lp->ppp_slot);
  return;
}
is = ippp_table[lp->ppp_slot];
isdn_ppp_frame_log('ccp-rcv',
  skb->data, skb->len, 32,
```

```
if (!lp->master)
  qdisc_reset(lp->netdev->
    dev.qdisc);
lp->dialstate = 0;
dev->st_netdev[isdn_dc2minor(
            lp->isdn_device
            lp->isdn_channel)
              ] = NULL;
isdn_free_channel(
    lp->isdn_device,
    lp->isdn_channel,
    ISDN_USAGE_NET);
lp->flags &=
  ISDN_NET_CONNECTED;
```

```
sidx = isdn_dc2minor(di, 1);
#ifdef ISDN_DEBUG_NET_ICALL
 printk(KERN_DEBUG "n_fi:ch=0\n");
#endif

if (USG_NONE(dev->usage[sidx])){
 if (dev->usage[sidx] &
      ISDN_USAGE_EXCLUSIVE) {
   printk(KERN_DEBUG "n_fi: 2nd
    channel is down and bound\n");
   if ((lp->pre_device == di) &&
       (lp->pre_channel == 1)) {
```

Determining Defect Report Similarity

- Some defect reports are obviously similar or different
- Some are not:

```
printk(KERN_DEBUG "Receive CCP
  frame from peer slot(%d)",
  lp->ppp_slot);
if (lp->ppp_slot < 0 ||
    lp->ppp_slot > ISDN_MAX) {
  printk(KERN_ERR "%s:
    lp->ppp_slot (%d) out of
    range", _FUNCTION_,
    lp->ppp_slot);
  return;
}
is = ippp_table[lp->ppp_slot];
isdn_ppp_frame_log('ccp-rcv',
  skb->data, skb->len, 32,
```

```
if (!lp->master)
  qdisc_reset(lp->netdev->
    dev.qdisc);
lp->dialstate = 0;
dev->st_netdev[isdn_dc2minor(
            lp->isdn_device
            lp->isdn_channel)
              ] = NULL;
isdn_free_channel(
    lp->isdn_device,
    lp->isdn_channel,
    ISDN_USAGE_NET);
lp->flags &=
  ISDN_NET_CONNECTED;
```

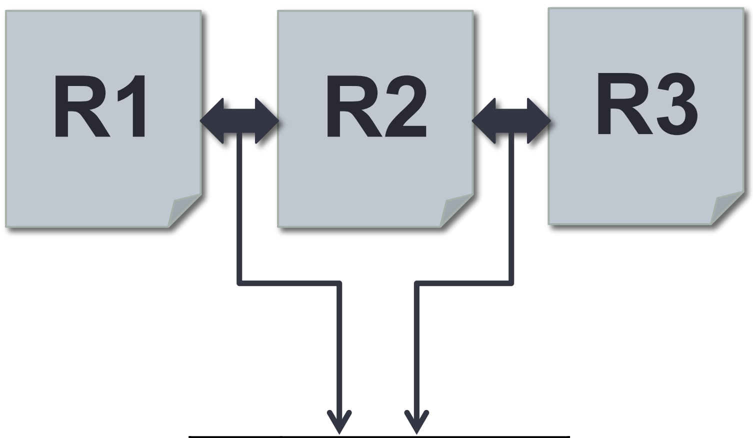
```
sidx = isdn_dc2minor(di, 1);
#ifdef ISDN_DEBUG_NET_ICALL
 printk(KERN_DEBUG "n_fi:ch=0\n");
#endif

if (USG_NONE(dev->usage[sidx])){
 if (dev->usage[sidx] &
       ISDN_USAGE_EXCLUSIVE) {
   printk(KERN_DEBUG "n_fi: 2nd
    channel is down and bound\n");
   if ((lp->pre_device == di) &&
       (lp->pre_channel == 1)) {
```

Goals

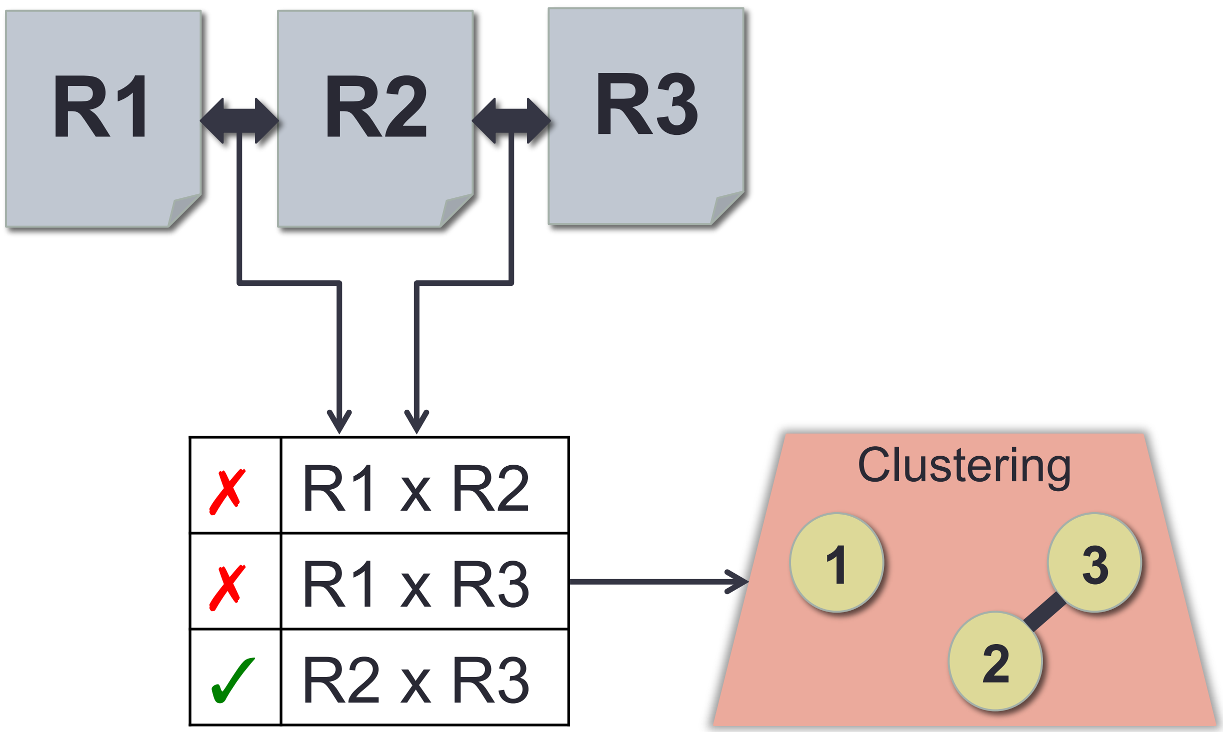
- To both aid in triage of real defects and facilitate the elimination of false positives, we desire a technique for clustering automatically-generated, static analysis-based defect reports.
- The technique should be flexible to meet the needs of different systems and development teams.
- The resulting clusters should be more accurate than those produced by existing baselines and also congruent with human notions of related defect reports.

High Level Approach

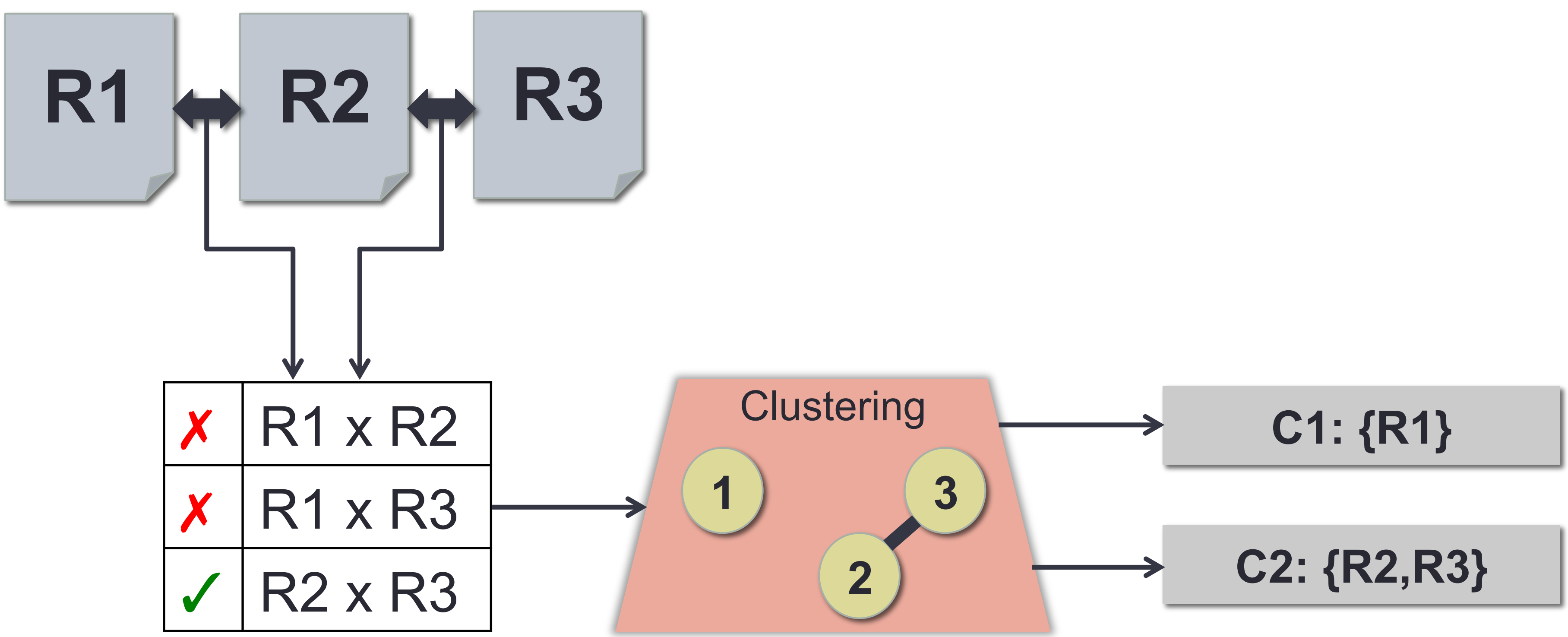


✗	R1 x R2
✗	R1 x R3
✓	R2 x R3

High Level Approach



High Level Approach



Approach – Types of Information

- Gathered or synthesized from structured defect reports
  - Type of defect
  - Suspected faulty line
  - Set of lines on static execution path to suspected fault
  - The enclosing function of the suspected fault
  - Three-line window of context around faulty line
  - Macros
  - File system path of suspected faulty file
  - Additional meta-information
- These categories conform to many state-of-the-art static analysis tools' output format
  - For instance, Coverity's SAVE tool and Findbugs

Approach – Types of Similarity Metrics

- Structured Similarity Metrics
  - Exact equality

```
Component comp = myGraph.subcomponent(size, false);
Component comp = g.subcomponent(getSize(), false);
```

Approach – Types of Similarity Metrics

- Structured Similarity Metrics
  - Exact equality
  - Strict pair-wise comparison

```
Component comp = myGraph.subcomponent(size, false);
Component comp = g.subcomponent(getSize(), false);
```

Approach – Types of Similarity Metrics

- Structured Similarity Metrics
  - Exact equality
  - Strict pair-wise comparison
  - Levenshtein edit distance

```
Component comp = myGraph.subcomponent(size, false);
Component comp = g.subcomponent(getSize(), false);
```

Approach – Types of Similarity Metrics

- Structured Similarity Metrics
  - Exact equality
  - Strict pair-wise comparison
  - Levenshtein edit distance
  - TF-IDF

```
Component comp = myGraph.subcomponent(size, false);

Component comp = g.subcomponent(getSize(), false);
```

Approach – Types of Similarity Metrics

- Structured Similarity Metrics
  - Exact equality
  - Strict pair-wise comparison
  - Levenshtein edit distance
  - TF-IDF
  - Largest common pair-wise prefix

```
Component comp = myGraph.subcomponent(size, false);
Component comp = g.subcomponent(getSize(), false);
```

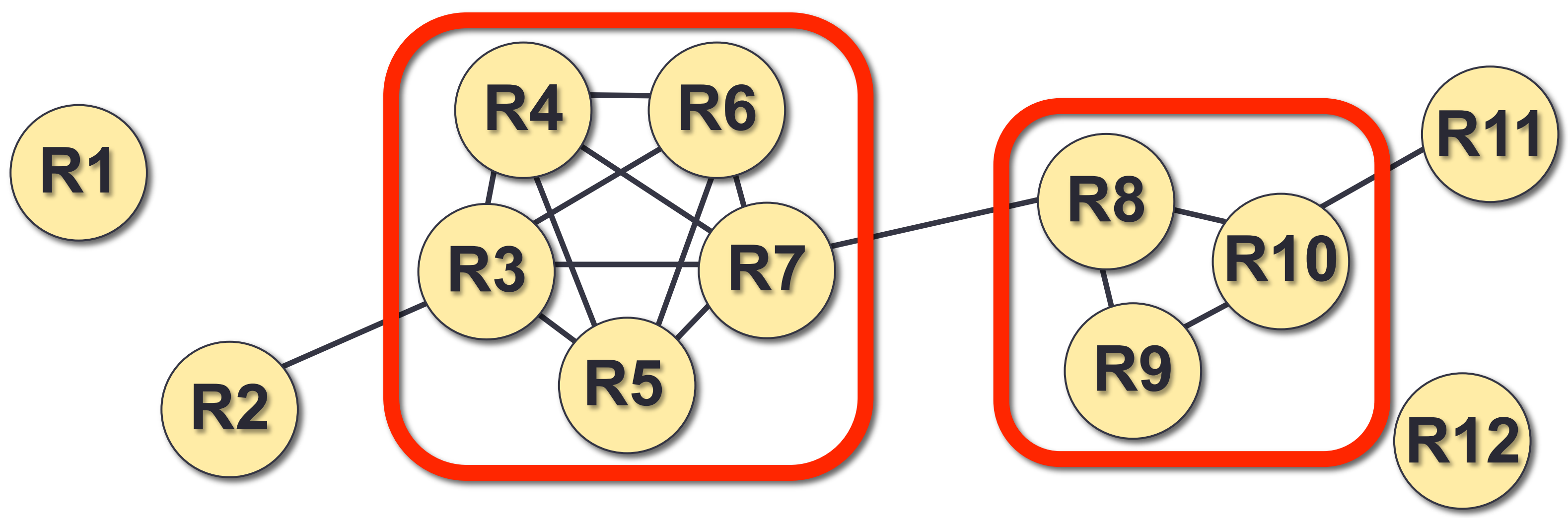
Approach – Types of Similarity Metrics

- Structured Similarity Metrics
  - Exact equality
  - Strict pair-wise comparison
  - Levenshtein edit distance
  - TF-IDF
  - Largest common pair-wise prefix
  - Punctuation edit distance

```
Component comp = myGraph.subcomponent(size, false);
Component comp = g.subcomponent(getSize(), false);
```

Approach – Similarity and Clusters

- Learn a linear regression model for all relevant information-metric pairs with similarity cutoff
- Traditional clustering (e.g. k-medoid) assumes equal feature weights and real-valued properties measured for individual entities
- Recursively find maximum cliques (clusters) and remove them from similarity graph



Evaluation

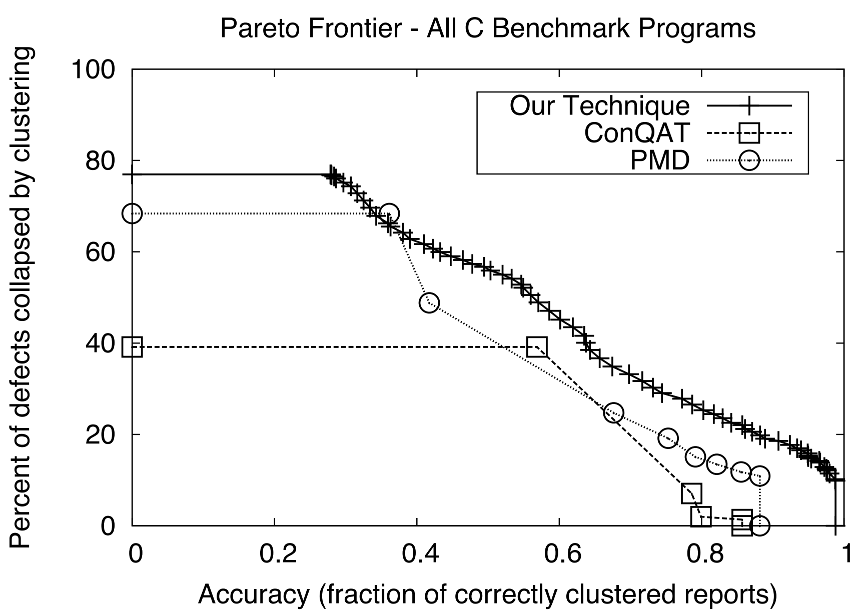
- Research Questions
  1. How effective is our technique at accurately clustering automatically-generated defect reports?
  2. Does our approach outperform existing baseline techniques?
  3. Do humans agree with the clusters produced by our technique?

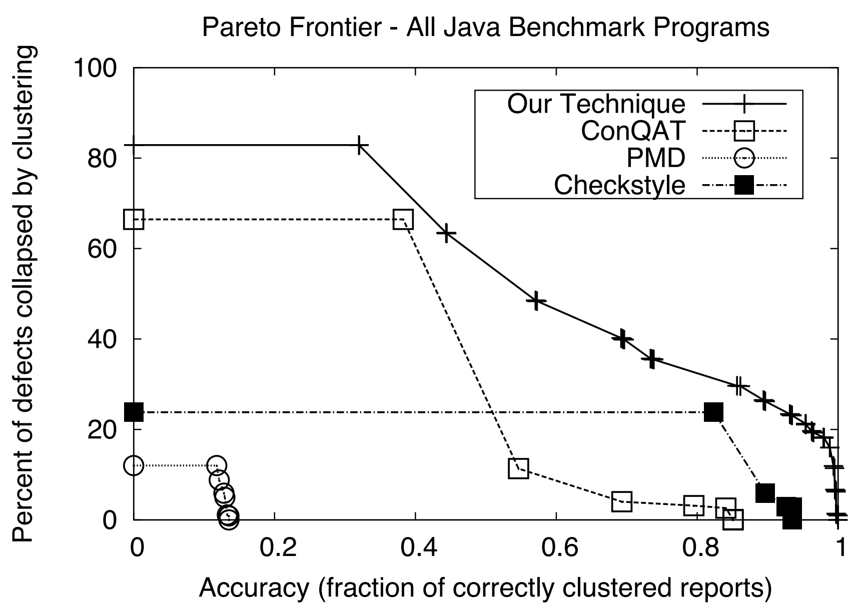
Evaluation

- Static analysis defect finding tools
  - Coverity SAVE (commercial) and Findbugs (open source)
- Benchmarks
  - Seven C and four Java open source programs totaling more than 14 million lines of code, yielding 8,948 defect reports
- Metrics – competing
  - Cluster accuracy
  - Cluster size
- Baseline techniques
  - Code Clone tools – Checkstyle, ConQAT, PMD
  - Well-established tools that solve a similar problem

Results

- Pareto frontier representing parametric choice between accuracy and cluster size
- Split between languages

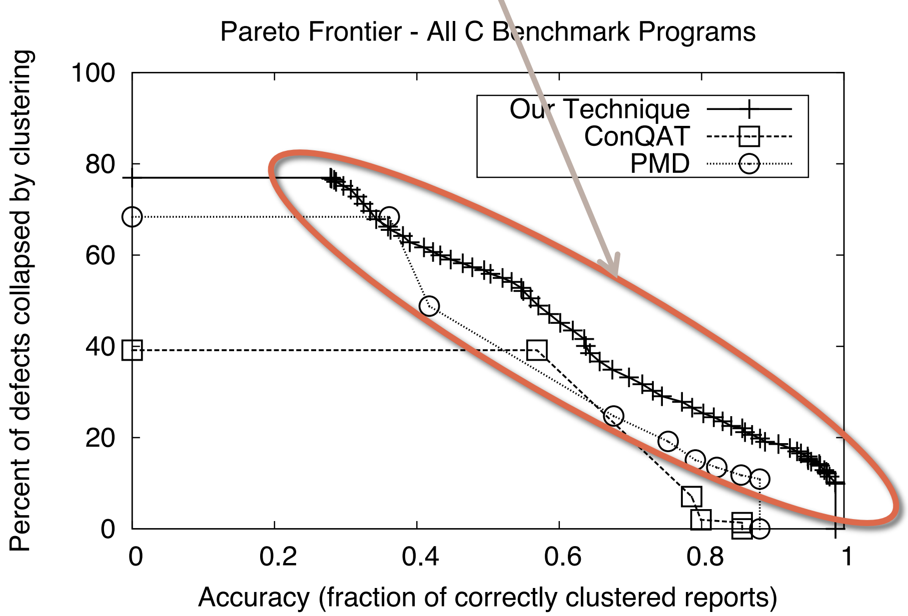


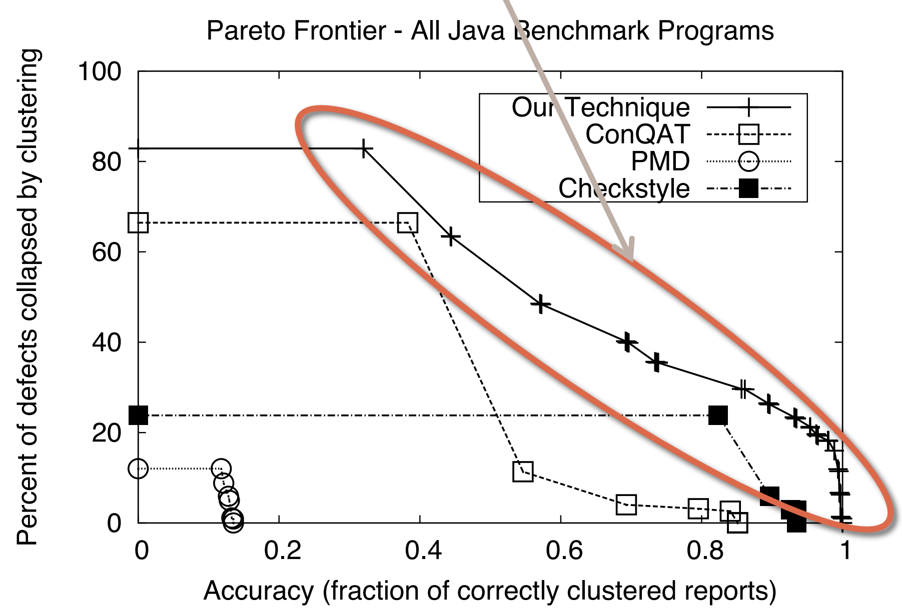


Results

Larger clusters at <u>most</u> levels of accuracy

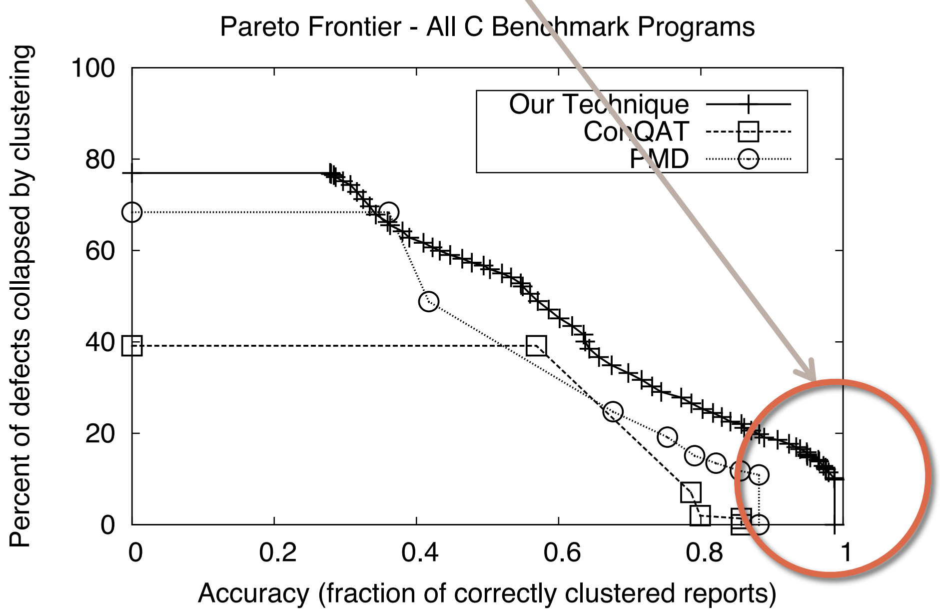
Larger clusters at <u>all</u> levels of accuracy



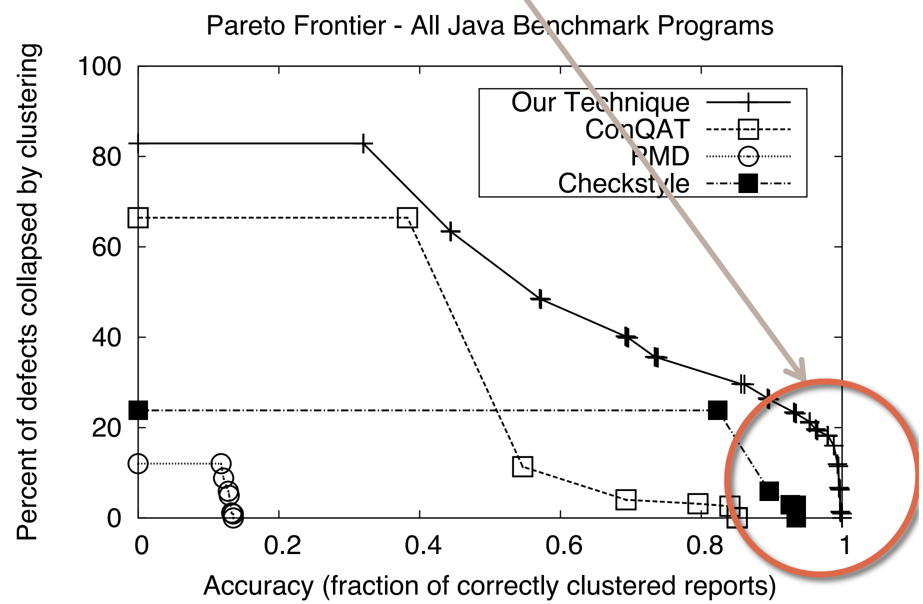


Results

Capable of <u>near perfect</u> accuracy



Capable of <u>perfect</u> accuracy



Cluster Quality

- Clusters ultimately should agree with humans' intuition of defect report similarity
- Given highly accurate (>90%) and highly inaccurate (<10%) clusters of actual defect reports, we asked humans if they thought the defect reports described the same or highly related bugs
- Results
  - "Accurate" clusters: 99% of humans think reports are related
  - "Inaccurate" clusters: 44% of humans think reports are related
- Humans do not overwhelmingly agree on inaccurate clusters
  - Motivates a parametric approach

Conclusion

- Defect reports from static analyses are prevalent and can be readily clustered.
- Our technique is effective at clustering such reports – it is capable of nearly perfect accuracy.
- Our technique outperforms the nearest baselines – with almost unanimously bigger clusters at all accuracy levels.
- Our technique produces accurate clusters – and humans agree with those clusters.