From Java to Python:

Migrating Search Functionality at billiger.de

Patrick Schemitz, solute GmbH
pycon.de 2017

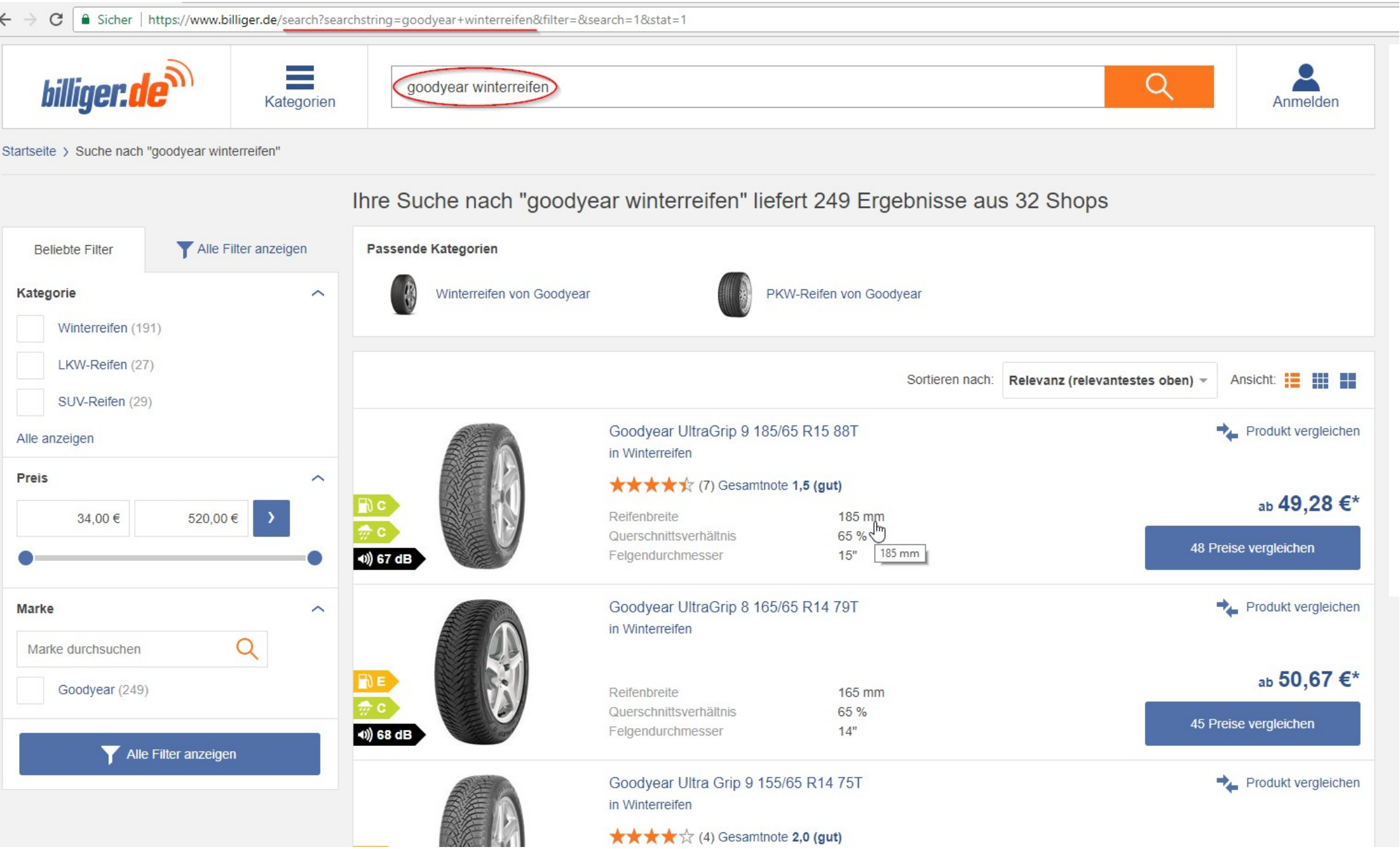
billiger.de

solute GmbH – billiger.de

- ca. 300 employees in Karlsruhe, Leipzig, Dresden and Plovdiv (Bulgaria)

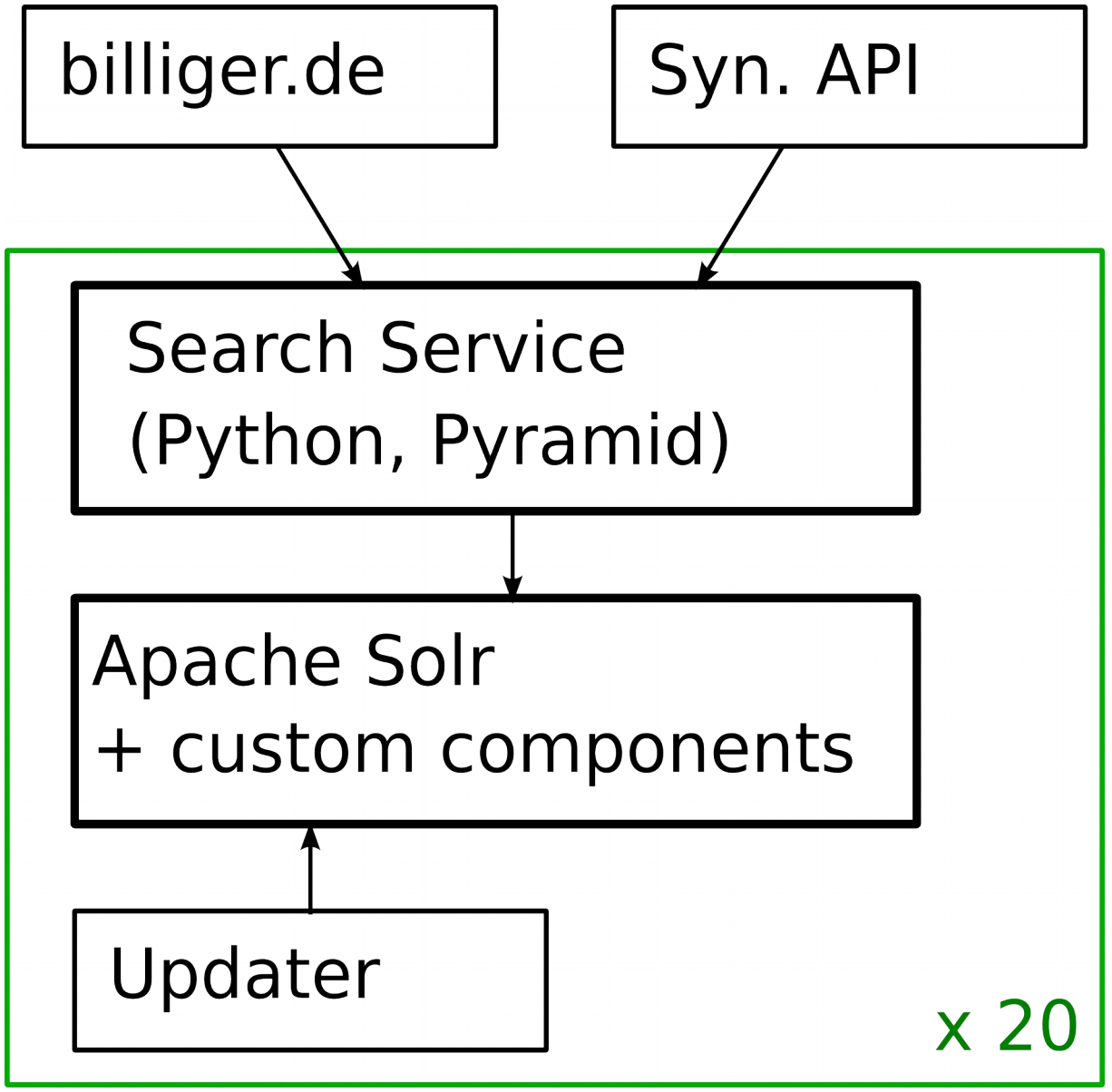
==> price comparison <==

- ***billiger.de*** and syndication partner dataset:
  - 2500 data feeds (from 50000+ dealers)
  - 65+ mio offers
  - volatility ø 6 mio offers/day
  - ø 200000 visitors/day (billiger.de)
  - ø 17 mio search requests/day (total)



■ 17M queries/day ■ 65M offers ■

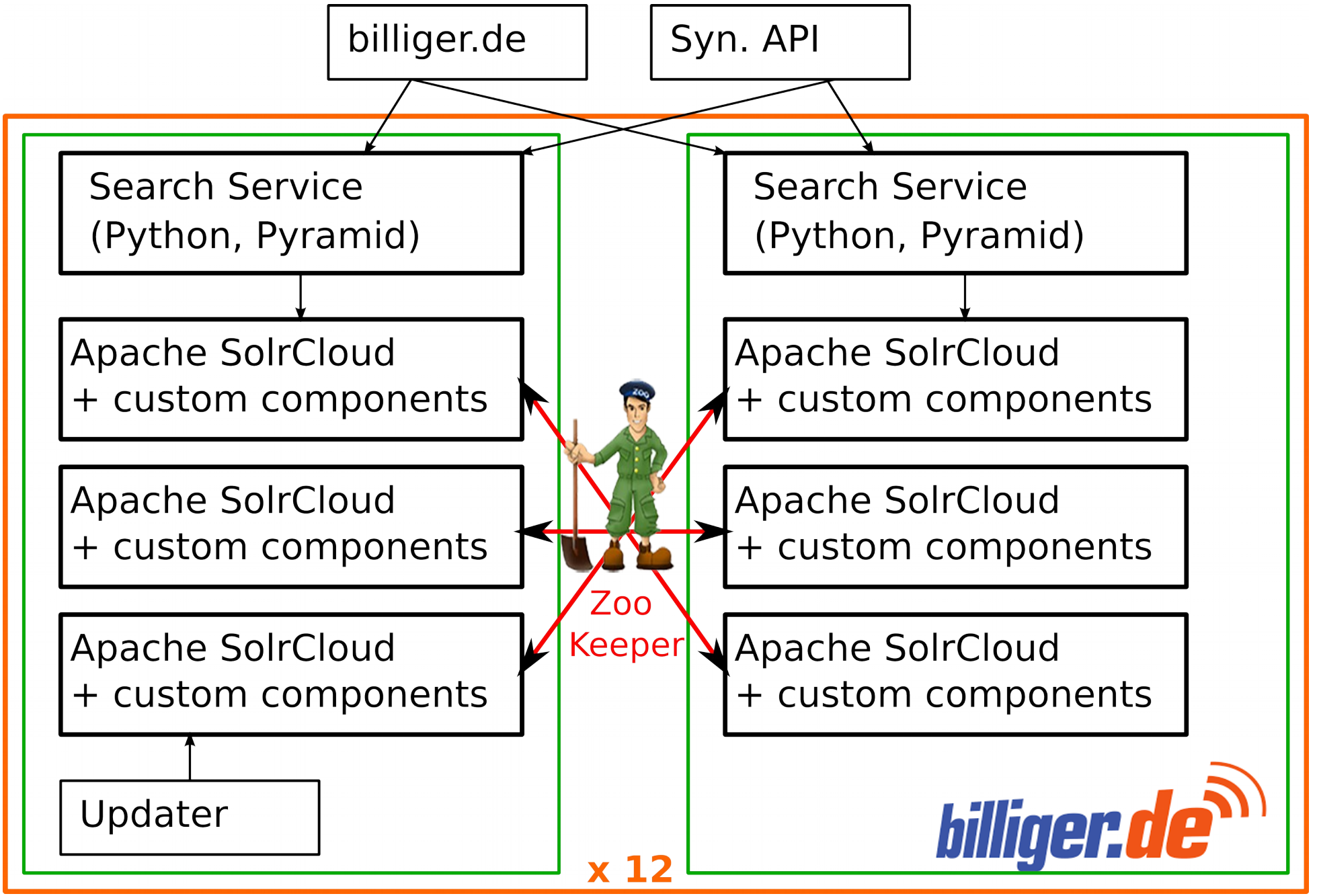
(Old) Architecture



- Dual 6-8 cores, 128 GB
- 1 Solr instance w/ *complete index*
- Search Service: *encapsulates actual search* (JSON/RPC, Python, Pyramid)

billiger.de

Target (Cluster) Architecture



Solr Components Interface

- Solr **`SearchComponent`** interface (*non*-cluster):

  Before actual search (but after query parsing):

```
public abstract void prepare(
    ResponseBuilder rb) throws IOException;
```

  After actual search:

```
public abstract void process(
    ResponseBuilder rb) throws IOException;
```

Custom Components I: QLTB

- Query-Local Term Boost: Boosting successful “terms” (offers, filters) for select queries:

```
<boosts>
  <query text="xperia x">
    <term boost="2.1" field="filter_brands"
          value="12316"/> <!--Sony-->
    <term boost="110.0" field="filter_categories"
          value="4373"/> <!--Handys ohne Vertrag-->
    <term boost="13000.0" field="id"
          value="1:808801927"/> <!--Xperia X 32GB black-->
  </query>
```

- Data harvested from clicklogs
- Updater puts QLTB.XML file into Solr config dir
- OSS: https://github.com/solute/qltb

billiger.de

QLTB Component

```
public class QLTBComponent extends SearchComponent {

  @Override
  public final void prepare(final ResponseBuilder rb) {
    Query query = rb.getQuery();
    String queryStr = rb.getQueryString();
    IndexReader reader = rb.req.getSearcher().getIndexReader();
    List<Query> boostTerms = getBoostsMap(reader,
                                    rb.req.getCore()).get(queryStr);
    BooleanQuery newq = new BooleanQuery(true);
    newq.add(query, BooleanClause.Occur.MUST);
    for (Query term : boostTerms) {
      newq.add(term, BooleanClause.Occur.SHOULD));
    }
    rb.setQuery(newq);
  }

  @Override
  public void process(final ResponseBuilder rb) {
  }
```

billiger.de

QLTB: Cluster Trouble

- Updater not on all nodes present
  → no qltb.xml?!
- SolrCloud: config distrib. by Apache ZooKeeper
- ZooKeeper config file size limit: 1 MB
  (need to recompile to adapt)
- We have a (company) standard way to distribute files: MogileFS
- Move functionality to Search Service!

billiger.de

QLTB Code: Python “prepare”

```
def search(..., query, ...):
    # ...
    qltb_terms = qltb.get_boost_terms(query)
    if qltb_terms:
        query_boosts = filters_to_solr(qltb_terms, with_boosts=True)
        for bq in query_boosts:
            solr_request.append(("bq", bq))
    # ...
```

billiger.de

QLTB Code: Python “prepare”

```
def search(..., query, ...):
    # ...
    qltb_terms = qltb.get_boost_terms(query)
    if qltb_terms:
        query_boosts = filters_to_solr(qltb_terms, with_boosts=True)
        for bq in query_boosts:
            solr_request.append(("bq", bq))
    # ...
```

Fits great into existing code for filters and boosts:

```
def search(..., query, ..., filters, boosts, ...):
    # ...
    query_filters = filters_to_solr(filters, with_boosts=False)
    for fq in query_filters:
        solr_request.append(("fq", fq))

    query_boosts = filters_to_solr(boosts, with_boosts=True)
    for bq in query_boosts:
        solr_request.append(("bq", bq))
    # ...
```

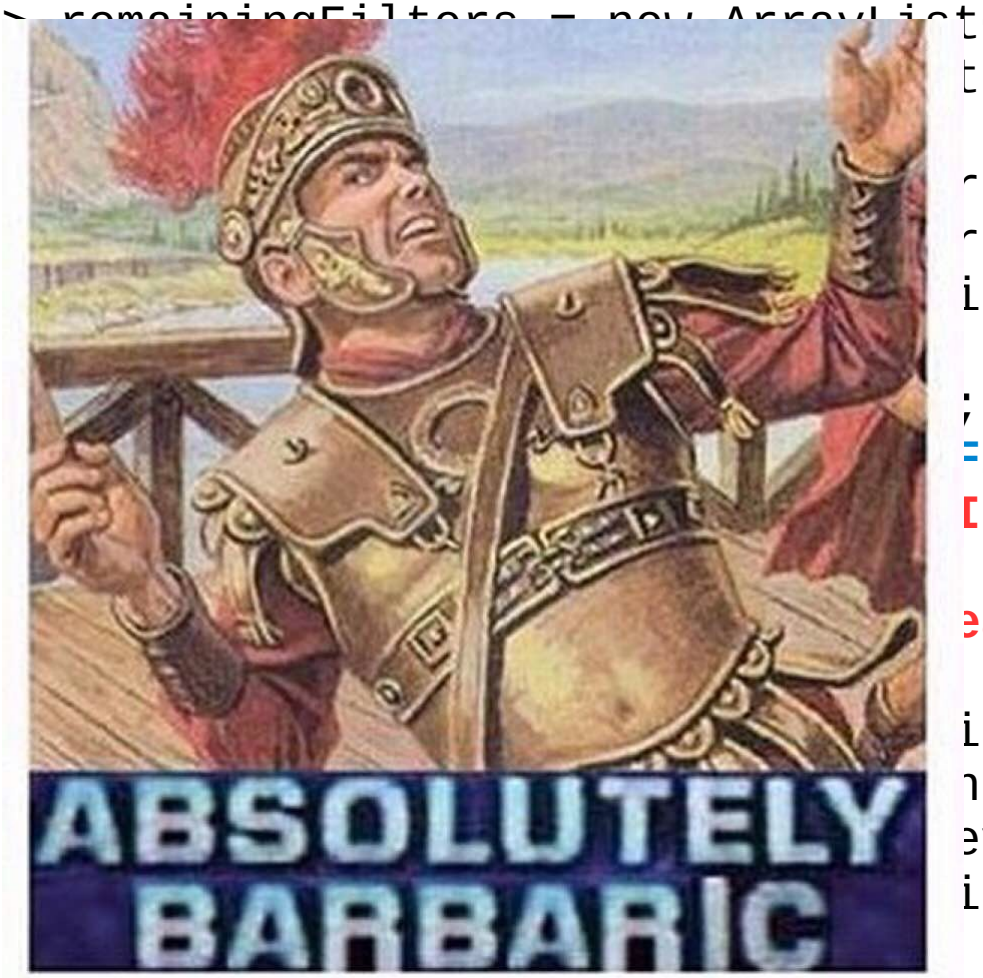
billiger.de

Custom Components II:
Facetting & Filter Alternatives

Facetting & Filter Alternatives

Filter Alternatives Component

```
@Override public final void process(final ResponseBuilder rb) {
  //...
  for (String filterField : termFilterFields) {
    List<Query> remainingFilters = new ArrayList<Query>();
    List<Strin                                    tring>();
    filterFiel
    recreateQu                                    rFields,
                                                  rs);
    List<Query                                    ist<Query>();
    alternativ
    alternativ                                    ;
    DocSet dcs                                    Filters);
    UnInverted                                    InvertedField(
        filter
    NamedList<                                    earcher, dcs, 0,
        limit,
    NamedList<                                    ist<Integer>();
    for (Entry                                    nts) {
        Strin                                     ey();
        Strin                                     it(":");
        if ((
            .equals(origFilterField)) {
             fieldCounts.add(keyValue[1],valuePair.getValue());
        }
```



Filter Alternatives Component

```
@Override public final void process(final ResponseBuilder rb) {
  //...
  for (String filterField : termFilterFields) {
    List<Query> remainingFilters = new ArrayList<Query>();
    List<String> filterFields = new ArrayList<String>();
    filterFields.add(filterField);
    recreateQueriesWithoutFields(filters, filterFields,
                                 remainingFilters);
    List<Query> alternativeFilters = new ArrayList<Query>();
    alternativeFilters.add(query);
    alternativeFilters.addAll(remainingFilters);
    DocSet dcs = searcher.getDocSet(alternativeFilters);
    UnInvertedField uif = UnInvertedField.getUnInvertedField(
        filterField, rb.req.getSearcher());
    NamedList<Integer> counts = uif.getCounts(searcher, dcs, 0,
        limit, 1, false, "count", null);
    NamedList<Integer> fieldCounts = new NamedList<Integer>();
    for (Entry<String, Integer> valuePair : counts) {
         String filterKeyValue = valuePair.getKey();
         String[] keyValue = filterKeyValue.split(":");
         if (("filter_" + keyValue[0])
             .equals(origFilterField)) {
              fieldCounts.add(keyValue[1],valuePair.getValue());
         }
```

Filter Alternatives: Cluster Trouble

- UnInvertedField dropped in Solr 6
- getDocSet() (=*search*) in a loop → latency!
- Move functionality to Search Service!

billiger.de

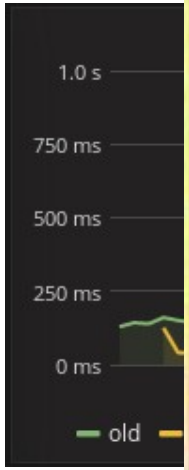
Filter Alternatives in Python

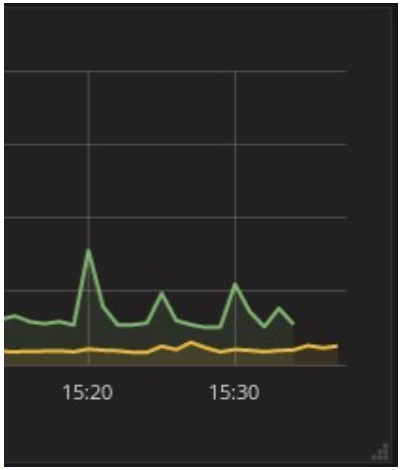
```
def search(index, query, page_size, page_no,
           sort_mode, filters, boosts, options, ...):
    # ...
    alternatives_requests = []
    for filter_key, filter_values in filters.iteritems():
        partial_filters = copy.deepcopy(filters)
        partial_filters.pop(filter_key)
        alternatives_requests.append((
            index, query,
            0, 0, # paging irrelevant here, 0 is fastest
            sort_mode,
            partial_filters,
            {}, # boosts irrelevant here, none are fastest
            options,
        ))

    alternatives_responses = search_threaded(
        alternatives_requests,
        _pool
    )
```

Conclusion

- Switch



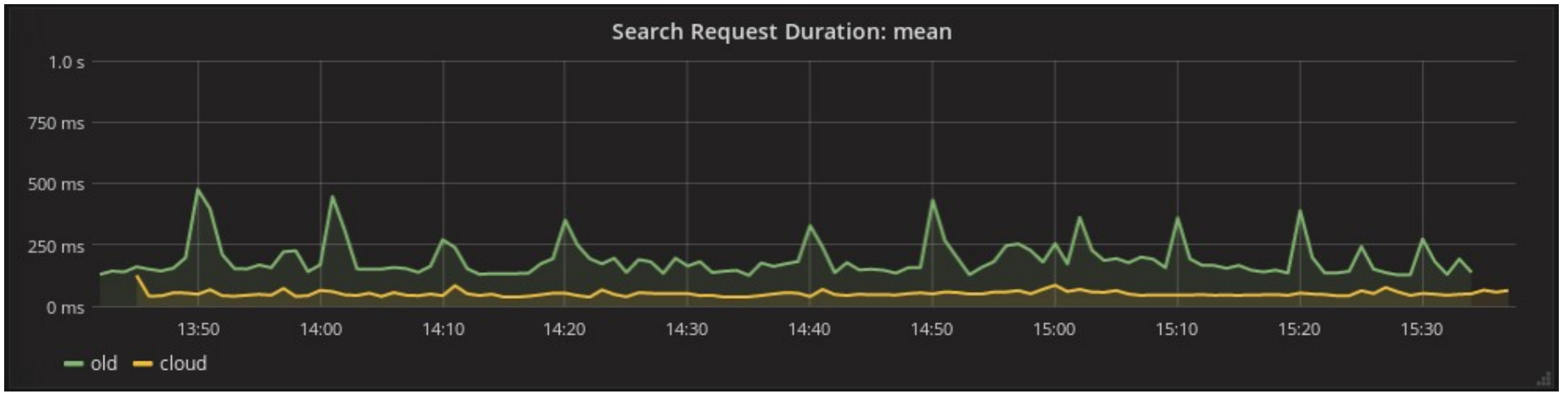


- SolrC
- Encap .I in migrating to Sol
- Move
- Pytho

billiger.de

Conclusion

- Switch from Solr to SolrCloud ✓

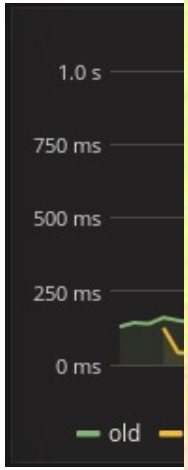


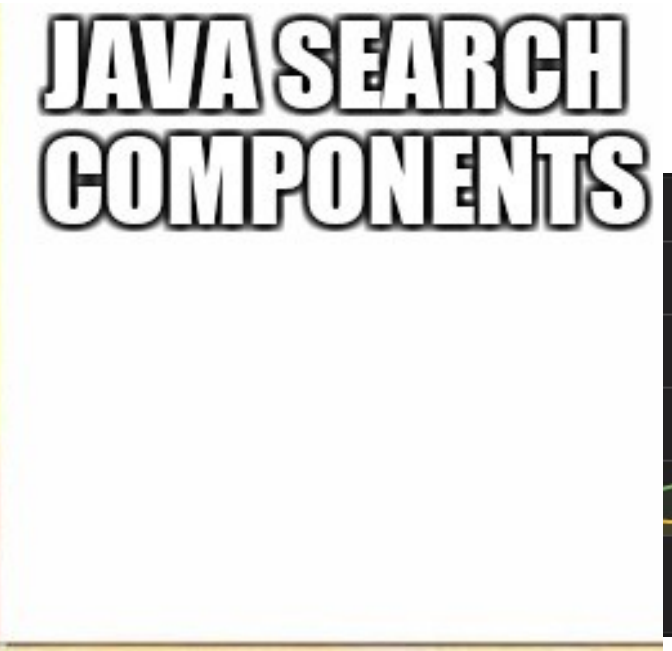
- SolrCloud works great ✓
- Encapsulating Solr in a (Python) service **crucial** in migrating to SolrCloud ✓
- Moved six SearchComponents to Python ✓
- Python is **awesome** ✓

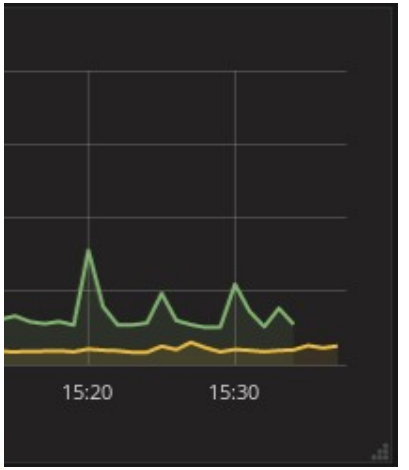
billiger.de

Conclusion

- Switch







- SolrC
- Encap .I in migrating to Sol
- Move
- Pytho

billiger.de

Minimum Match Python Code

```
def search(..., query, ...):
    # ...
    solr_request.append(("mm", "100%"))
    # ...
    result = solr_search(url, solr_request, shop_fq, options)
    # ...
    if result["total_hits"] == 0:
        _replace_param(solr_request, "mm", "75%")
        result = solr_search(url, solr_request, shop_fq, options)
```

billiger.de

Minimum Match: Cluster Trouble

- prepare() executed on each shard/instance:
  - getDocSet() (= *search*) on each shard:
    → **mm might vary between shards**
    Some shards would return precise hits, others imprecise

- Move functionality to Search Service!

Minimum Match Component

```
public class FuzzyComponent extends SearchComponent {

    @Override public void prepare(ResponseBuilder rb) {
        String queryString = rb.getQueryString();
        SolrParams params = rb.req.getParams();
        Query q = rb.getQuery();
        List<Query> filters = rb.getFilters();
        List<Query> queries = new ArrayList<Query>();
        queries.add(q);
        queries.addAll(filters);
        DocSet result = rb.req.getSearcher().getDocSet(queries);
        if (result.size() > 0)
            return;
        q = QParser.getParser("{!mm=75%} " + queryString,
                              defType, rb.req).getQuery();
        queries = new ArrayList<Query>();
        queries.add(q);
        queries.addAll(filters);
        DocSet result = rb.req.getSearcher().getDocSet(queries);
        if (result.size() > 0)
            rb.setQuery(q);
    }
```

Custom Components III: "Minimum Match" (fuzzy search)

- "Minimum Match" parameter: number (or percentage) of words in a query that need to match a doc.
- Example: query "apple samsung galaxy s7"
  - mm=100% (*all* terms must match) yields no hits
  - mm=75% (3 terms must match) yields "samsung galaxy s7" but also **random stuff** (if any): "apple samsung s7", "apple galaxy s7" ...
  - Strategy:
    - try mm=100% first
    - try mm=75% only if no mm=100% hits

billiger.de