



Berner Fachhochschule

Projektarbeit CAS Big Data, 2016

(library) - [:DATA] -> (service)

– Weiterentwicklung ausgewählter Komponenten der Informationsplattform swissbib als
Grundlage für Dienstleistungen wissenschaftlicher Bibliotheken im Jahre 2020+ –

vorgelegt von:

Student:	Günter Hipler
Weiterbildung:	CAS Big Data, Frühjahr 2016
Telefon-Nr.:	061 3 82 82 29
E-Mail:	guenter.hipler@unibas.ch

Basel, den 9.10.2016

Inhalt

1	MANAGEMENT SUMMARY.....	1
2	EINLEITUNG.....	2
2.1	BESCHREIBUNG DES PROJEKTUMFELDS.....	2
2.2	VORGEHENSWEISE UND ARBEITSPAKETE DER PROJEKTARBEIT.....	4
3	ARCHITEKTUR DER AKTUELLEN SWISSBIB PLATTFORM.....	6
3.1	ARCHITEKTURBESCHREIBUNG.....	6
3.1.1	<i>Grundgedanken beim Aufbau.....</i>	<i>6</i>
3.1.2	<i>Konsequenzen und Erfahrungen aus der Architekturentscheidung.....</i>	<i>7</i>
3.2	PROBLEME UND ANSATZPUNKTE FÜR DIE WEITERENTWICKLUNG.....	8
3.2.1	<i>Analyse von Datenflüssen auf der swissbib Plattform.....</i>	<i>9</i>
3.2.2	<i>Analyse der Komplexität und des Funktionsumfangs einzelner swissbib Plattform Komponenten.....</i>	<i>13</i>
3.2.2.1	Kohäsion der aktuellen data ingest Komponente.....	13
3.2.2.2	Kohäsion der document processing Komponente für die SOLR Suchmaschine.....	14
3.2.2.3	Kohäsion des aktuellen Datenhubs.....	15
3.2.2.4	Kopplung von Datenhub für traditionelle Clusteringverfahren mit der Datenmanagementplattform für verlinkte Daten.....	15
3.2.3	<i>Skalierbarkeit einzelner aktueller Komponenten.....</i>	<i>16</i>
3.2.3.1	Skalierbarkeit des Datenhubs.....	16
3.2.3.2	Skalierbarkeit des Data Ingestions.....	17
4	LAMBDA / KAPPA ARCHITEKTUR ALS MÖGLICHE BASIS UND ERSTER SCHRITT HIN ZU EINER ZUKÜNFTIGEN SWISSBIB PLATTFORM.....	17
5	EVALUATION / IMPLEMENTIERUNG EINZELNER KOMPONENTEN DER NEUEN WEITERENTWICKELTEN SWISSBIB PLATTFORM IM RAHMEN VON PROTOTYPEN.....	22
5.1	APACHE KAFKA ALS EVENT HUB.....	22
5.1.1	<i>Installation von Kafka.....</i>	<i>22</i>
5.1.2	<i>Entwicklung eines Konsumenten für den Kanal OAI.....</i>	<i>23</i>
5.1.3	<i>Erste Beurteilung des Prototyps und next steps.....</i>	<i>25</i>
5.2	METAFACURE CLUSTER ALS KOMPONENTE EINES BATCH LAYERS.....	26
5.2.1	<i>Einführung in Metafacture und Metafacture_cluster.....</i>	<i>26</i>
5.2.2	<i>Benutztes Setup (Infrastruktur) für das Arbeitspaket 3.....</i>	<i>30</i>
5.2.3	<i>Laden eines ersten Testdatensets mit Kafka als Datenquelle.....</i>	<i>31</i>
5.2.4	<i>Erläuterung des entstandenen spalten-orientierten Datenmodells unter Verwendung von Metamorph.....</i>	<i>35</i>
5.2.5	<i>Erste Beurteilung des Prototyps und next steps.....</i>	<i>37</i>
5.3	ERSTE EVALUATION VON STREAMPROZESSOREN FÜR EIN SCHLANKES UND MODERNES DATA PROCESSING AUF DER SWISSBIB PLATTFORM.....	40
5.3.1	<i>Prototyp: Anbindung von Kafka in Apache Flink.....</i>	<i>44</i>
5.3.2	<i>Prototyp: Anbindung von HBase in Apache Flink.....</i>	<i>45</i>
6	FAZIT / AUSBLICK.....	45
7	LITERATURVERZEICHNIS.....	48

Verzeichnis der Abbildungen

Abb. 1: Layered Architecture swissbib.....	7
Abb. 2: aktuelle erweiterte auf Layern basierte swissbib Architektur.....	8
Abb. 3: Daten- und Komponentenintegration ohne einen Event-Hub.....	11
Abb. 4: Daten- und Komponentenintegration ohne einen Event-Hub.....	12
Abb. 5: Infrastruktur auf Basis der Lambda / Kappa Architektur.....	18
Abb. 6: Installation Kafka / Zookeeper.....	23
Abb. 7: Abhängigkeiten der Python Umgebung.....	24
Abb. 8: Klassendiagramm Kafka Produzenten.....	24
Abb. 9: einfacher workflow bestehend aus reader, Transformation und writer. Die Transformation führt ein Metamorph Modul auf Basis einer übergebenen DSL Definition durch.....	27
Abb. 10: InputTableMapper als Komponente für MetafactureWorkflows und Hadoop Mapper mit Anschluss an ein HBase Tabelle für den Bezug von Daten.....	28
Abb. 11: Setup der Infrastruktur zur Umsetzung von Arbeitspaket 2.....	30
Abb. 12: Komponenten für das Laden von Daten aus Kafka nach HBase.....	31
Abb. 13: Illustration wie Metafacture-Workflows zusammengebaut werden können.....	33
Abb. 14: Ergebnis einer Metamorphtransformation im Formeta-Format.....	34
Abb. 15: Auszug eines rows aus der HBase Ladetabelle.....	35
Abb. 16: Schemaübersicht der workflows für das Culturegraph-Portal.....	38
Abb. 17: work-cluster im ElasticsearchIndex nach Aggregation durch Spark.....	41
Abb. 18: schlanke data processing Lösung der swissbib Plattform im Zusammenspiel und in Integration mit SLSP.....	42

1 Management Summary

Das Projekt swissbib, eine öffentliche Serviceplattform für Daten und Suchdienste im Umfeld Schweizer wissenschaftlicher Bibliotheken, hat eine seiner Kernkompetenzen im mittlerweile 6-jährigen Betrieb und Management eines nationalen Datenhubs aufgebaut. Dieser Datenhub, der die durch swissbib aufbereiteten Daten (und damit den Rohstoff) von mehr als 900 bibliothekarischen Institutionen enthält, ist die Grundlage für Services und Schnittstellen der swissbib Plattform, die von vielen Menschen aber auch maschinellen clients genutzt wird.

Der Titel der Arbeit (library) - [:DATA] -> (service) greift diese Erfahrung auf und drückt den allgemeinen Gedanke aus, dass Services von (wissenschaftlichen) Bibliotheken in Zukunft ohne innovative und moderne Verfahren und Methoden des data processing kaum kompetitiv erbracht werden können.

Die Geschwindigkeit sowie der Innovationsgrad der OpenSource ‚BigData‘ Entwicklung der letzten 10 Jahre hat stabile Komponenten entstehen lassen, die auch nicht spezialisierte Softwareunternehmen wie öffentliche Institutionen im Bibliotheksbereich in die Lage versetzen, diese neuen Werkzeuge für ihre Zwecke einzusetzen.

Die Mischung aus OpenSource als auch kommerzieller Software sowie das flexible Design des swissbib Service ermöglichten regelmässige Anpassungen der Architektur von swissbib an Umweltveränderungen bereits innerhalb der letzten 6 Jahre und sind eine sehr gute Grundlage für die Adaption dieser modernen Verfahren.

Die vorliegende Arbeit analysiert die aktuelle Architektur der swissbib Plattform und entwickelt eine Zielarchitektur mit ausgewählten Komponenten der heutigen BigData Entwicklungen.

Es werden einzelne Prototypen von neuen Komponenten dieser Zielarchitektur für ein Proof of Concept entwickelt. Diese Prototypen greifen u.a. eine Implementierung der Deutschen Nationalbibliothek aus dem Jahre 2013 auf und versuchen die damalige Initiative mit den aktuellen (neuen) Möglichkeiten in Verbindung zu setzen. Es wird gezeigt, dass die Zielarchitektur kein Big-Bang eines komplett neuen Systems sein muss sondern ihr Wert gerade in der Möglichkeit der Integration von Teilen der bewährten aktuellen Plattform mit den neuen Möglichkeiten besteht. Die komponentenorientierte Architektur mit offenen Schnittstellen ermöglicht es sehr gut, die swissbib Plattform

mit ähnlichen Initiativen in der Schweiz zu kombinieren.

2 Einleitung

2.1 Beschreibung des Projektumfelds

Das Projekt swissbib an der Universitätsbibliothek Basel ist eine im Rahmen des Programms SUK-P2¹ durch swissuniversities² geförderte öffentliche Serviceplattform für Daten- und Suchdienste. Die bibliographischen Metadaten aller schweizerischen Universitätsbibliotheken, der Nationalbibliothek, einzelner Kantonalbibliotheken, diverser Dokumentenrepositories sowie zusätzlichen Spezialsammlungen z.B. aus Archiven werden in einem Datenhub aufbereitet³. Dieser Datenhub ist Grundlage für interaktive Services (Information Retrieval durch Forschende, Studenten und die interessierte Öffentlichkeit)⁴ oder APIs⁵, die wiederum von anderen Services für ihre Zwecke genutzt werden können. Durch die Aufbereitung im Datenhub werden ca. 30 Millionen Datensätze der einzelnen Institutionen mittels Deduplizierungsmechanismen auf ca. 21 Millionen Aufnahmen zusammengefasst. Diese 21 Millionen Entities werden über Clusteringverfahren (Zusammenfassung von ähnlichen Aufnahmen wie zum Beispiel mehrere Auflagen eines Werkes) sowie Verlinkung und Anreicherung mit externen Objekten nochmals veredelt.

Diese traditionellen, von Bibliotheken selbst produzierten, bibliographischen Beschreibungen, werden in Zukunft durch andere Formen von Metadaten zumindest im Volumen überholt werden:

- Metadaten von digitalen Objekten (vor allem Artikeln)⁶
- weniger stark strukturierte Beschreibungen von Forschungsdaten

¹ Für weitere Informationen vgl. <https://www.swissuniversities.ch/de/organisation/projekte-und-programme/suk-p-2-wissensch-information-zugang-verarbeitung-speicherung/>
https://www.swissuniversities.ch/fileadmin/swissuniversities/Dokumente/DE/UH/SUK_P-2/Abstract_swissbib.pdf

² <https://www.swissuniversities.ch>

³ Für eine Zusammenstellung der beteiligten Institutionen: <https://www.swissbib.ch/Libraries>

⁴ <https://www.swissbib.ch>

⁵ z.B. <http://sru.swissbib.ch>

⁶ vgl. die aktuelle Zusammenarbeit mit dem Projekt Nationallizenzen
<http://swissbib.blogspot.ch/2016/03/national-licences-and-article-metadata.html>

- Metadaten aus anderen kulturellen Einrichtungen als Bibliotheken (GLAM)

Mit diesen unterschiedlichen Anforderungen muss eine moderne swissbib Datenplattform umgehen können.

Der Titel der Projektarbeit (library) - [:DATA] → (service) möchte die Bindegliedfunktion von Daten zwischen Institutionen und den von ihnen angebotenen Diensten zum Ausdruck bringen. Hierfür wurde die Syntax der Abfragesprache Cypher⁷ der populären Graphendatenbank Neo4J gewählt, deren Einsatzschwerpunkt vor allem in der Vernetzung von Entitäten zu sehen ist. Nach Auffassung des Autors werden Institutionen in Zukunft nur dann in der Lage sein Services, die den wechselnden Bedürfnissen und Wünschen von BenutzerInnen gerecht werden, anzubieten, wenn sie es schaffen, eine moderne Infrastruktur für das Management und die Verarbeitung von Daten als das Bindeglied zwischen Institutionen und Services aufzubauen und ständig zu erneuern. Während der Titel der Arbeit den Charakter eines Konzepts ausdrücken soll, ist das Netzwerk innerhalb der von swissbib angebotenen Daten viel umfangreicher. Einen ersten Eindruck kann ein kleines Projekt des swissbib Teams am „Open Cultural Data Hackathon 2016“ der Universitätsbibliothek Basel geben. Hier wurde versucht, die Beziehungen innerhalb der auch von swissbib verwendeten Autoritätsdaten zu visualisieren⁸

Wenn das Management der Daten von so zentraler Bedeutung für die anzubietenden Services von Institutionen ist, was heisst das dann für die notwendige Infrastruktur, auf der das Management stattfinden kann? Auf diese Frage versucht die Projektarbeit Antworten und Empfehlungen zu geben. Es wird aufgezeigt, wie sich die Serviceplattform swissbib weiterentwickeln muss, um den Anforderungen an eine Informationsinfrastruktur im Jahre 2020+ gerecht werden zu können. Die Lösung orientiert sich an folgenden Rahmenbedingungen:

- Es soll eine evolutionäre Entwicklung sein, die auf den Grundprinzipien der aktuellen Architektur aufbauen kann bzw. in der Lage ist, vorhandene und bewährte Komponenten zu integrieren. Kein „Big-Bang“ eines neuen Systems.

⁷ <https://neo4j.com/developer/cypher-query-language/>

⁸ http://make.opendata.ch/wiki/project:visualize_relationships

- Das neue System muss in der Lage sein, sehr grosse Datenmengen mit hoher Veränderungsrate und unterschiedlichsten Strukturen zu verarbeiten⁹. Dabei wird davon ausgegangen, dass Bibliotheken auch in Zukunft sich die Möglichkeit bewahren, die von ihnen aufwendig selbst erstellten und/oder mit hohem finanziellen Aufwand beschafften Daten vollumfänglich selbst bewirtschaften zu können. Dies auch im abzusehenden verstärkten „Gang in die Cloud“ – was dies auch immer konkret bedeuten wird. Vollumfänglich selbst bewirtschaften heisst für den Autor, im Besitz der ursprünglichen Rohdaten zu sein und nicht nur durch zum Beispiel angebotene APIs kommerzieller Cloudbetreiber bereits veränderte und/oder aggregierte Sekundärdaten beziehen zu können.
- Die neue Infrastruktur muss nicht nur flexibel und skalierbar mit heterogenen, hohen Datenmengen umgehen können, sie muss auch agil auf neue Benutzer- und damit Serviceanforderungen reagieren können. Es reicht heute nicht mehr, sog. Funktionskataloge für Systeme in der Hoffnung zu erstellen, diese die nächsten 5 bis 10 Jahre ohne Veränderung einfach fortschreiben zu können.

2.2 Vorgehensweise und Arbeitspakete der Projektarbeit

Arbeitspaket 1: Architekturbeschreibungen, -überlegungen

Zuerst (Kapitel 3.1) wird die aktuelle Architektur der swissbib Plattform vorgestellt. Wir werden erkennen, wie sich die Lösung evolutionär weiter entwickeln kann und welche bestehenden Komponenten in eine neue Lösung einfliessen können. Im Kapitel 3.2 werden die aktuell erkannten Schwächen erläutert und diskutiert.

Kapitel 4 stellt eine erste neue Architektur vor, beschreibt und diskutiert sie. Es wird aufgezeigt, wie die in Kapitel 2.1 aufgestellten Rahmenbedingungen der zukünftigen Lösung erfüllt werden könnten.

Im eigentlichen Hauptteil der Arbeit sollen zwei Komponenten evaluiert und prototypmässig implementiert werden. Arbeitstechnisch und zur einfacheren Kommunikation mit Lesern dieser Arbeit, habe ich dafür 2 Arbeitspakete vorgesehen:

⁹ Vgl. hierzu die „5 V“ genannten Charakteristiken von Big Data Lösungen
https://en.wikipedia.org/wiki/Big_data

Arbeitspaket 2: Integration eines Event Hub / Neuentwicklung der data ingestion Komponente

Im Rahmen dieses Paktes arbeite ich mich in die Möglichkeiten eines Eventhubs auf Basis von Apache Kafka ein und erstelle den ersten Prototypen eines Producers als Teil des zukünftigen data ingestion (Kapitel 5.1).

Arbeitspaket 3: Evaluation von metafacture_cluster als Batch MapReduce Framework zur Analyse von Metadaten

Metafacture_cluster könnte als batchorientierte Komponente für eine sog. Batch-view innerhalb einer Lambda-Architektur eingesetzt werden. Einzelheiten dazu im Kapitel 5.2

Arbeitspaket 4: Einstieg und erste Überlegungen zum Einsatz von Streamprozessoren als Kern einer schlanken und gleichzeitig modernen data processing Plattform

Da sich in den letzten 2 – 3 Jahren zunehmend Data-Analytics Streamprozessoren als Ersatz für batchorientierte Verfahren empfohlen haben und diese sehr gut mit einem Kafka Eventhub kombinierbar sind, werde ich in einem Abschlusskapitel 5.3, quasi als Zusatz zu den ursprünglich zwei vorgesehenen Hauptthemen, beschreiben, zu welchem Ergebnis mich meine Einarbeitungsschritte und ersten Implementierungen mit dieser Technologie geführt haben. Ohne den Inhalten der Kapitel vorzugreifen sei hier bereits kurz erwähnt, dass Streamingtechnologien, wie sie in den Apache Projekten Flink und Spark¹⁰ entwickelt werden, wohl die technologische Basis für einen Metadatenhub der Zukunft sein sollten. Die sich daraus ergebende, im Vergleich zu Kapitel 4 erweiterte, Zielarchitektur wird ebenfalls erläutert.

Mit der prototypmässigen Umsetzung einzelner Komponenten und Teile einer zukünftigen Plattform im Rahmen dieser Projektarbeit möchte ich auch aufzeigen, dass die aufgezählten Architekturpattern und (OpenSource) Softwarelösungen heute nicht mehr nur Thema für spezialisierte Softwareunternehmen sind, sondern einen Reifungsprozess durchlaufen haben und sich dadurch in einem Status befinden, der es vertretbar erscheinen lässt, sie auch im breiteren Rahmen in wissenschaftlichen Bibliotheken einsetzen zu können. Für mich ist diese Entwicklung vergleichbar mit den OpenSource

¹⁰ Vgl. <https://flink.apache.org> und <http://spark.apache.org/>

Suchmaschinen Solr und ElasticSearch und der darunter liegenden Lucene library¹¹, die sich innerhalb der letzten 5 – 10 Jahre behauptet haben und heutzutage allgegenwärtig sind, sei es durch Eigenentwicklungen von Bibliotheken oder durch die (versteckte) Nutzung in kommerziellen Discovery Lösungen. Meine persönliche Einschätzung: (OpenSource) BigData Analytics Lösungen durchlaufen derzeit zeitversetzt den gleichen Zyklus.

3 Architektur der aktuellen swissbib Plattform

3.1 Architekturbeschreibung

3.1.1 Grundgedanken beim Aufbau

Der Aufbau der aktuellen swissbib Plattform, die sich seit Februar 2010 im produktiven Einsatz befindet, folgt einem konsequenten Architekturmuster. Die eingesetzten Komponenten

- Data ingestion und pre-processing (content collection)
- Data management und enrichment (data hub)
- Document processing und Search engine
- Komponenten für Benutzer- und Maschinenservices

sind voneinander unabhängig in einzelnen Layern angeordnet und kommunizieren ausschliesslich über Schnittstellen miteinander. Diese Schnittstellen sind nicht nur intern verfügbar sondern werden grösstenteils auch externen Services bereitgestellt. Dies ermöglichte im Verlaufe der Zeit den Austausch einzelner Komponenten, nachdem sich die äusseren Umstände verändert hatten. So wurde aus einer nahezu 100% kommerziellen Lösung eine Plattform, die zu einem grösseren Teil auf Open Source Komponenten beruht¹². Der Leitgedanke von swissbib ist nicht von einem sich gegenseitig ausschliessenden “make or buy” sondern von einem sich je nach den Umständen ergebenden “make and buy” geprägt. Dies machte den service in einer sich permanent verändernden Umwelt sehr flexibel und agil.

¹¹ Vgl. <https://lucene.apache.org/core/>
<http://lucene.apache.org/solr/>
<https://www.elastic.co/products/elasticsearch>

¹² s. auch <https://github.com/swissbib/> und <https://github.com/linked-swissbib> Ausgetauscht wurden vor allem die layer der Suchmaschine (von Fast zu SOLR), der Präsentationskomponente (kommerzielles Produkt zu <http://vufind-org.github.io/vufind/> sowie der ingest Komponente)

Dieses ursprüngliche Design ist in dem nachfolgenden Schaubild abgebildet. Es wurde bereits im Jahre 2013 im Rahmen einer Präsentation vorgestellt¹³

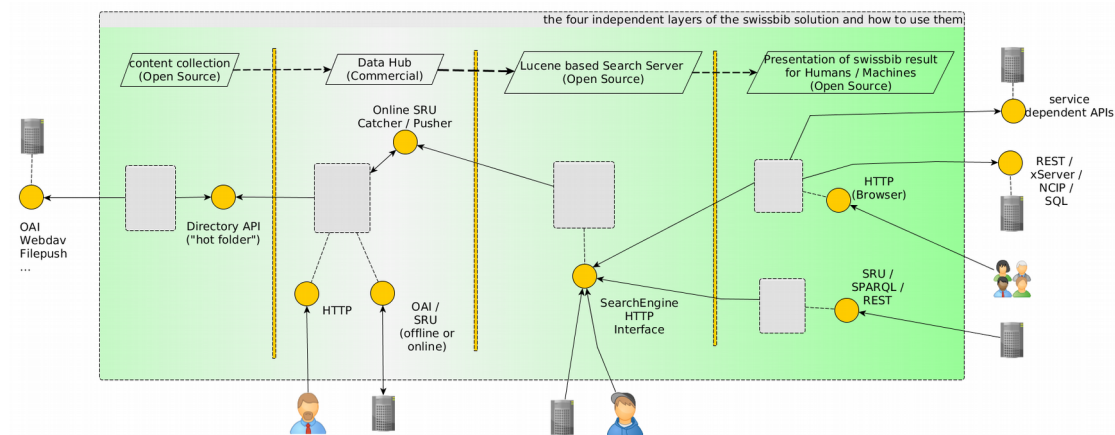


Abb. 1: Layered Architecture swissbib
Quelle: eigene Darstellung

Um den Rahmen dieser Arbeit nicht zu sprengen, verzichte ich an dieser Stelle auf weitergehende Details zur Basisarchitektur. Interessierte Personen finden zusätzliche Informationen im Manuskript zur angegebenen Präsentation.

3.1.2 Konsequenzen und Erfahrungen aus der Architekturentscheidung

Diese Anpassbarkeit bewährte sich auch in der letzten grösseren Erweiterung seit Beginn des Jahres 2015 im Rahmen des SUK-P2 Projekts linked-swissbib¹⁴. Es mussten Komponenten gefunden werden, die den bestehenden Datenhub um die Möglichkeiten erweiterten, bibliographische Metadaten in das RDF Format zu transformieren und die so entstehenden neuen Ressourcen mit externen Ressourcen zu verlinken. Hierfür verwenden wir das an der Deutschen Nationalbibliothek im Rahmen des Projekts Culturegraph entwickelte Framework Metafacture¹⁵. Es baut auf einer datenflussorientierten Domain Specific Language (Metamorph) auf und benutzt Pattern zur Datenanalyse, die denen von Hadoop mit MapReduce ähnlich sind. Dadurch ist eine Transformation hin zu einem batchorientierten Datencluster möglich und wurde im Projekt Me-

¹³ vgl. <http://swissbib.blogspot.ch/2013/12/overview-and-advantages-of-layered.html>

¹⁴ http://campus.hesge.ch/id_bilingue/projekte/linkedswissbibch/default.asp

15 <https://github.com/culturegraph/metafacture-core>

tafacture-cluster¹⁶ auch umgesetzt. Bisher jedoch nur im Rahmen eines Prototyps. Die nachfolgende Abbildung verdeutlicht die Integration der linked-data Plattform.

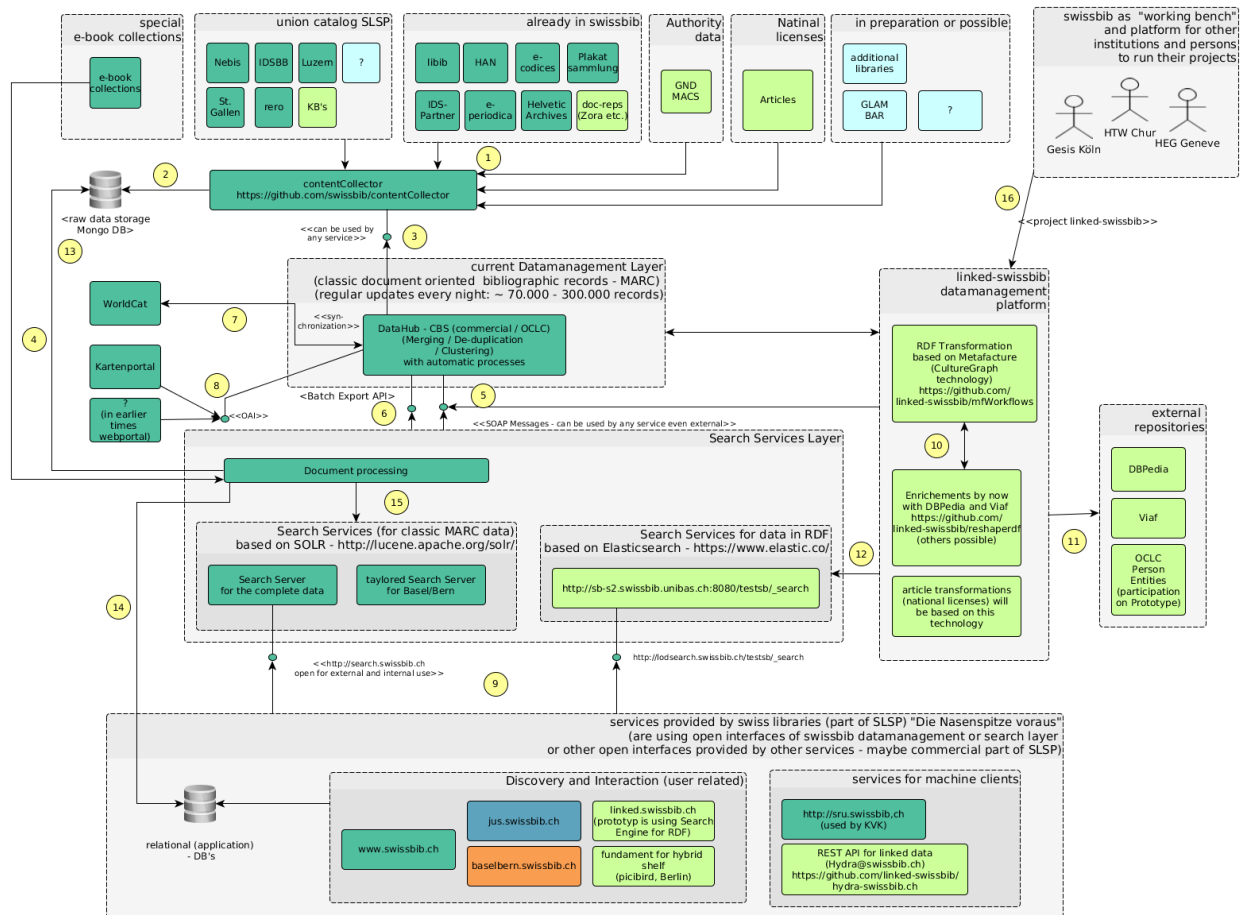


Abb. 2: aktuelle erweiterte auf Layern basierte swissbib Architektur
Quelle: eigene Abbildung

3.2 Probleme und Ansatzpunkte für die Weiterentwicklung

Das Grundprinzip eines gelayerten Systems mit darin befindlichen Komponenten, die über offene Schnittstellen miteinander kommunizieren, hat sich bewährt, auch im permanenten Ausbau. Vergleicht man Abb. 1 und Abb. 2 miteinander, erkennt man auf den ersten Blick, dass die Architektur durch die Weiterentwicklung zumindest unübersichtlicher geworden ist. Gründe hierfür sind u.a.

- die Heterogenität und der Umfang der Daten hat zugenommen

¹⁶ <https://github.com/culturegraph/metafacture-cluster>

- während zu Beginn des Projekts der Datenfluss gleichmässig und in eine Richtung verlaufen ist (data ingestion liefert an Datenhub, Datenhub stellt content dem document processing bereit, Präsentationsoberfläche -discovery- benutzt die Suchmaschinenschnittstelle und liefert den BenutzerInnen Daten) ist heute die wechselseitige Abhängigkeit zwischen einzelnen Komponenten des Daten-systems grösser. Eine Folge davon war auch die Zunahme des Funktionsumfangs und damit Komplexität einzelner Komponenten.

3.2.1 Analyse von Datenflüssen auf der swissbib Plattform

Gwen Shapira beschreibt in [1]¹⁷ den Hauptverwendungszweck für den Event Hub Apache Kafka: Technisch ein sehr schnelles, verteiltes, hoch skalierbares und verfügbares Publish/Subscribe messaging System löst es auf der konzeptionellen Ebene ein bisher eher schwierig zu lösendes Problem: Die Integration und Kommunikation von Softwarekomponenten innerhalb eines (Daten)-Systems. Vor dem Hintergrund dieser Aussage möchte ich nun die Datenflüsse der aktuellen swissbib Architektur kurz beschreiben. Hierzu beziehe ich mich auf einzelne Ziffern am Rand von Schnittstellen oder Kommunikationslinien der Abb. 2.

- data ingestion (1,2,3,4): Hier findet ein wichtiger und vom Volumen her grosser Datenaustausch statt. Heterogene Daten werden über unterschiedliche Kanäle (OAI Protokoll, WebDav, File-Push) aus einer Vielzahl von Datenquellen eingesammelt (1) und nach einer minimalen Validierung in dem raw storage einer NoSQL Datenbank (MongoDB) abgelegt (2). Das Format der eingesammelten Daten aber auch die Formen des Ingestion können sich in Zukunft erheblich erweitern. Neben anderen bibliographischen Formaten sind bspw. interaktionsbezogene Daten von BenutzerInnen denkbar, mit denen die Qualität von angebotenen Services verbessert werden kann. Nicht zuletzt interaktive Daten sind für Streamingszenarien sehr geeignet. Die im Rahmen des Ingestions erhaltenen Daten werden zu einem Grossteil unserem aktuellen Datenhub über eine Fileschnittstelle bereitgestellt (3). Ein Beispiel für eine solche Ausnahme ist ein spezieller eBook-Pool der direkt in das Document-Processing unserer Suchmaschine überführt wird (4). Solche „Ausnahmefälle“ kann man sich in Zukunft häufiger vorstellen.

¹⁷ Vgl. „The case for Kafka“

- Exportschnittstellen des Datenhubs (5,6,7,8): Der Datenhub kennt verschiedene Schnittstellen, mit denen er anderen Komponenten seinen Inhalt zur Verfügung stellt. Sehr wichtig sind für uns SOAP- und Batch-Export Schnittstellen (5,6), über die einzelne Messages oder ein kompletter Export dem document processing unserer Suchmaschine für klassische Marc-Daten (SOLR, 15) oder der Aufbereitung von RDF Daten in der Komponente für Linked-Data bereitgestellt werden. Externe Systeme (ausserhalb der swissbib Plattform) verwenden eine OAI Schnittstelle (8). Daneben gib es noch einen proprietären Kanal (7), den der Hersteller des Datenhubs für den Austausch der Daten in beiden Richtungen mit dem weltweiten Worldcat verwendet.¹⁸
- Innerhalb der Komponente für linked-data werden Daten auf verschiedenen Ebenen aufbereitet und zusammengeführt (10), es werden Rohdaten für Anreicherungs Zwecke benötigt (11) und das Ergebnis des Prozesses in einer Suchmaschine (ElasticSearch) indiziert (12)
- Das document processing benötigt zur Anreicherung der Suchmaschinendokumente Daten, die zum einen aus Autoritätsdateien stammen (GND der Deutschen Nationalbibliothek) (13) oder dem Inhalt von geparsten Volltexten¹⁹ (14).
- Direkte Benutzersysteme (heute vor allem discovery) oder maschinelle clients (SRU oder REST für linked-data) verwenden die HTTP basierten Queryschnittstellen der Suchmaschinen (9)
- Wir haben bei uns mittlerweile auch den use case integriert, dass externe Institutionen die Plattform swissbib direkt für ihre Projekte nutzen (16)²⁰. Während dies aktuell noch auf Projekte mit Institutionen beschränkt ist, kann man sich hier in Zukunft auch direkte „data mining“ use cases für BenutzerInnen vorstellen. Hier denken wir zum Beispiel an den Einsatz des Apache Zeppelin Tools.²¹ Andere use cases für den direkteren (auch maschinellen) Zugriff auf Daten sind im Rahmen des Semantic Web möglich. Bisher verzichten wir (nach einer kurzen Testphase) auf den Einsatz von sog. Sparql Servern, weil die Skalierbarkeit auf grosse Datenmengen nicht unseren Vorstellungen und Erwar-

¹⁸ Vgl. <https://www.oclc.org/worldcat.en.html>

¹⁹ Aktuell beschränken wir uns auf TOC („table of content“) und Abstracts von Monographien.

²⁰ Kooperationsprojekt linked-swissbib zusammen mit der HTW Chur (<http://www.htwchur.ch/>), HEG Geneve (<https://www.hesge.ch/heg/>) und der Gesis in Köln (<http://www.gesis.org/das-institut/adresse-und-anreise/standort-koeln/>)

²¹ Vgl. <https://zeppelin.apache.org/>

tungen entsprochen hat. Das kann sich in Zukunft ändern. Zum Beispiel werden wir den Einsatz des Linked Data Fragments Frameworks²² testen. Damit würden wir noch offenere Alternativen zum bisherigen „Serviceweg“ bieten, bei dem Daten ausschliesslich über Services bereitgestellt werden und die manche deswegen auch als sogenannte Datensilos bezeichnen.

Für mich ist es interessant, die Resultate einer rund siebenjährigen (Weiter)-Entwicklungszeit eines data-processing Systems wie swissbib mit den Darstellungen in der allgemeinen Literatur zu vergleichen. Man findet viele Beschreibungen, wie zusätzliche Anforderungen und wachsende Datenmengen zu höherer Komplexität führten und dies die Betroffenen darüber nachdenken liess, wie man diesen Anforderungen begegnen könnte. In [1] beschreibt Gwen Shapira diese Vorgänge und verdeutlicht dies mit den zwei Abbildungen 3 und 4, die das Zusammenspiel von Datenflüssen und daran beteiligten Services allgemein in Unternehmen illustrieren sollen.

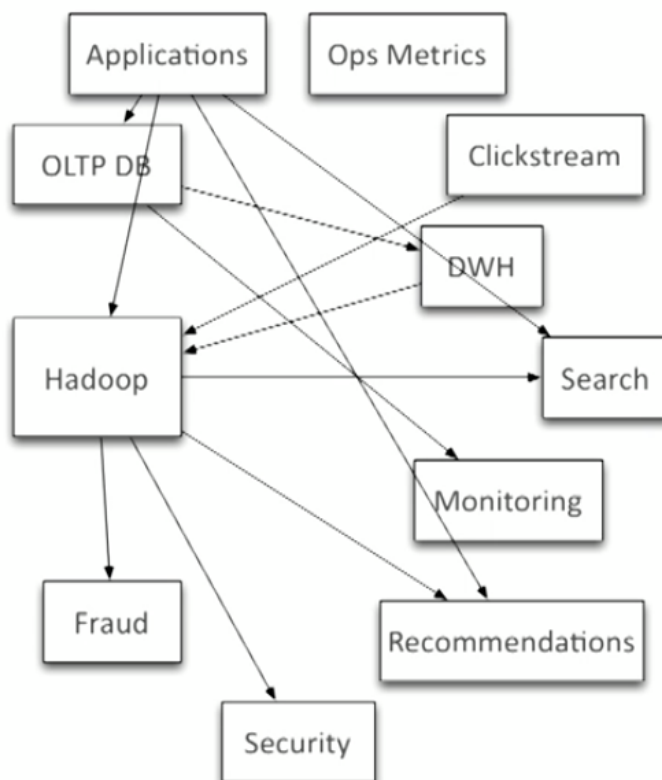


Abb. 3: Daten- und Komponentenintegration ohne einen Event-Hub
Quellenhinweis: Gwen Shapira [1]

²² Vgl. <http://linkeddatafragments.org/>

Als Lösung für die sichtbar erhöhte Komplexität von Datenflüssen und wachsende Abhängigkeit von Services (die der Abb. 2 in meinen Augen nicht ganz unähnlich ist) wird der Einsatz eines sogenannten Event Hubs empfohlen, wie er von Apache Kafka implementiert wird. Bei Gwen Shapira sieht die Lösung dann wie folgt aus:

The World with Kafka

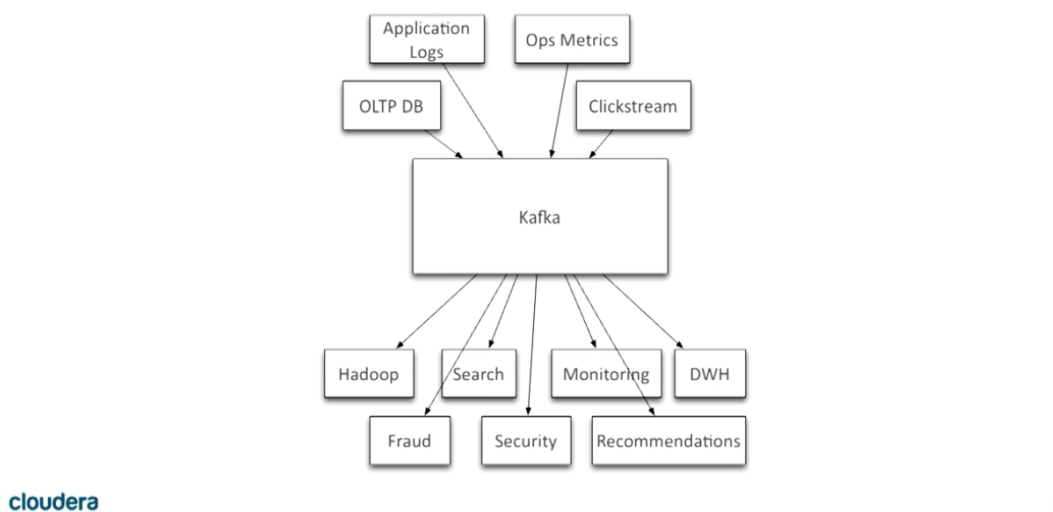


Abb. 4: Daten- und Komponentenintegration ohne einen Event-Hub
Quellenhinweis: Gwen Shapira [1]

Jay Kreps behandelt in [2] ähnliche (ausführlichere) Integrationsszenarien auf Basis eines dort noch „Unified Log“ genannten Systems, dem ursprünglichen Konzept für einen Event Hub ²³.

Wie die Lösung auf der zukünftigen swissbib Plattform aussehen kann, werde ich im 4. Kapitel beschreiben.

²³ Die Geschichte von Apache Kafka ist bekannt. Die Fa. LinkedIn war mit den von mir beschriebenen Anforderungen im data-processing konfrontiert und startete die Entwicklung der Basis für das heutige Apache Kafka. Das ursprüngliche Entwicklungsteam bei LinkedIn gründete um die OpenSource Software eine eigene Fa. mit dem Namen confluent vgl. <https://www.linkedin.com/company/confluent>

3.2.2 Analyse der Komplexität und des Funktionsumfangs einzelner swissbib Plattform Komponenten

Steigt die Vernetzung einzelner Komponenten innerhalb eines Systems (vgl. vorherigen Abschnitt) ist dies in der Regel auch mit einer erhöhten Komplexität der Implementierung einer Komponente verbunden. Diese muss in der Lage sein, mehrere Aspekte, die im Zusammenhang mit der Kommunikation anderer Systeme stehen, umzusetzen. Im Softwareengineering werden diese Zusammenhänge unter den Begriffen Kohäsion²⁴ und Kopplung²⁵ beschrieben. Wie sich im Laufe der mittlerweile sechsjährigen Produktionszeit die Kohäsion einzelner Komponenten aufgeweicht hat und Komponenten stärker gekoppelt wurden als grundsätzlich wünschenswert, möchte ich an drei Beispielen aufzeigen.

3.2.2.1 Kohäsion der aktuellen data ingest Komponente²⁶

Zu Anfang des Projektes entwickelten wir ein Modul, welches die folgenden Funktionen umsetzen sollte:

- Standardfunktionen eines OAI clients (Pull-Verfahren)
- Verarbeitung von Files, welche im Push Verfahren zur swissbib Plattform geschickt wurden.
- Bereitstellung des neuen content in einer (proprietären) File-Schnittstelle für unseren Datenhub

Im Laufe der Zeit kamen dann immer mehr Funktionen hinzu wie zum Beispiel:

- Ablage der Daten in einem Rohdatenspeicher (Mongo – DB) vor Übergabe an den Datenhub
- Unterstützung neuer Protokolle und workflows (z.B. WebDav). Durch die weiter wachsende Heterogenität unserer Datenquellen wird diese Anforderung auch in Zukunft bestehen bleiben.
- aufwendigere Validierungen der Daten. Wir mussten erleben, dass unser Datenhub, wenn auch nur in sehr wenigen Fällen, die Verarbeitung eines kompletten Batchjobs abbrach, weil Daten inkonsistent gewesen sind. Aus diesem Grund verlegten wir pre-validations in die ingestion Komponente, da Erweiterungen in der für uns offenen Software leichter umzusetzen sind.

²⁴ Vgl. [https://de.wikipedia.org/wiki/Koh%C3%A4sion_\(Informatik\)](https://de.wikipedia.org/wiki/Koh%C3%A4sion_(Informatik))

²⁵ Vgl. [https://de.wikipedia.org/wiki/Kopplung_\(Softwareentwicklung\)](https://de.wikipedia.org/wiki/Kopplung_(Softwareentwicklung))

²⁶ Vgl. <https://github.com/swissbib/contentCollector>

- Unterstützung von Ausnahmefällen einzelner Datenquellen trotz an sich standardisierter Protokolle. Einzelne Datenquellen erweiterten das Protokoll aus diversen Gründen, was bei uns ebenfalls Erweiterungen auslöste.
- Versuch über Hashmethoden von der Datenquelle nach unseren Kriterien doppelt gesendete messages auszufiltern
- wachsende Funktionalität war in unserem Fall mit zunehmender Konfiguration verbunden.

Trotz eines sehr nützlichen Plugin-Mechanismus, mit dem zusätzliche Funktionalität über definierte Schnittstellen mittels Konfigurationen eingehängt werden kann²⁷ und zweimaliger Refaktorisierung innerhalb von 6 Jahren, hätten wir uns die Möglichkeiten eines Event-Hubs wie Kafka, der erst Ende 2012 zu einem top -level Apache Projekt avancierte und innerhalb der letzten zwei Jahre enorm an Popularität gewonnen hat, gewünscht. Dadurch hätte die wünschenswerte Kohäsion weniger stark nachgelassen.

3.2.2.2 Kohäsion der document processing Komponente²⁸ für die SOLR Suchmaschine

Das swissbib document processing für die SOLR Suchmaschine folgt einem Pipeline – Prinzip, wie wir es bei der kommerziellen Suchmaschine FAST²⁹ kennengelernt hatten. Miteinander verkettete XSLT Templates³⁰ werden zu einer pipe zusammengefasst und rufen für komplexere Aufgaben Java Plugins auf. Diese Plugins entsprechen den stages in der früheren FAST Architektur. Obwohl der Plugin Mechanismus recht mächtig und vielseitig einsetzbar ist, verleitet er doch dazu, Aufgaben einer Komponente zuzuweisen, die in einer stärker lose gekoppelten Architektur besser an anderer Stelle implementiert werden sollten. Ein gutes Beispiel hierfür ist das Plugin zur Anreicherung von Suchdokumenten mit definierten Inhaltsverzeichnissen oder Abstracts³¹, mit dem dieser content innerhalb des document processing abgerufen, mit Tika³² geparkt und die extrahierten Terme anschliessend in einem data storage zur wiederholten Nutzung ab-

²⁷ Vgl. <https://github.com/swissbib/contentCollector/blob/master/harvestingTasks.py#L21>

²⁸ Vgl. <https://github.com/swissbib/content2SearchDocs>

²⁹ Vgl. https://de.wikipedia.org/wiki/Fast_Search_and_Transfer Das Unternehmen FAST wurde Ende 2008 (nach unserem Projektstart im April 2008) von Microsoft übernehmen. 2012 ersetzten wir FAST durch SOLR.

³⁰ Vgl. <https://github.com/swissbib/content2SearchDocs/tree/master/xslt>

³¹ Vgl. <https://github.com/swissbib/content2SearchDocs/blob/master/src/java/org/swissbib/document-processing/plugins/FulltextContentEnrichment.java>

³² <https://tika.apache.org/>

gelegt werden. Dieser Aspekt liesse sich leicht auslagern, wenn dafür ein eigener, einfach zu implementierender Konsument eines Event Hubs geschrieben wird, der diese Aufgabe genau dann übernimmt, sobald content im Rahmen des data ingestion neu in die Plattform aufgenommen und sofort einem Event-Hub übergeben wird.

3.2.2.3 Kohäsion des aktuellen Datenhubs

Aus den Abb. 1 und 2 lässt sich die Herkunft des klassischen Datenhub als universelle und generelle Serviceplattform für Mensch und Maschine erkennen. Obgleich wir ihn innerhalb unseres Projekts mehrheitlich als batchorientierte Komponente für Deduplizierungs- und Clusteringverfahren verwenden, bietet er zusätzliche maschinelle (OAI / SRU / SOAP) sowie interaktive Schnittstellen (HTTP) an. Ebenso ist die Synchronisation von Daten mit externen Plattformen³³ möglich. Heute würde man solche Services eher in einen eigenen Servicelayer auslagern und die Komponente für das data processing von solchen Aufgaben befreien.

3.2.2.4 Kopplung von Datenhub für traditionelle Clusteringverfahren mit der Datenmanagementplattform für verlinkte Daten

In der Entwicklung der Datenmanagementplattform für verlinkte Daten haben wir momentan die Schwierigkeit, dass wir den klassischen Datenhub als Basis benutzen. Die Verfahren basieren jedoch auf sehr unterschiedlichen Technologien und haben nicht deckungsgleiche Zielsetzungen. Der klassische Datenhub ist darauf spezialisiert, Informationen aus vielen verschiedenen Quellen zusammenzuführen und das Resultat in erster Linie für die interaktive Benutzersuche bereitzustellen. In diesem Prozess ist es nicht so wesentlich, wenn ursprüngliche Informationsbündel aufgrund definierter Regeln später wieder auseinandergenommen und BenutzerInnen in neuer Zusammensetzung präsentiert werden. Ressourcen im „Web of Data“ sollten jedoch, wenn immer möglich, unter einer persistenten Adresse abrufbar sein, was zumindest für bibliographische Ressourcen nicht immer möglich sein wird, wenn man das erste Verfahren als Grundlage verwendet. Der Widerspruch wird sich in Zukunft ein wenig entspannen, wenn wissenschaftliche Bibliotheken in der Schweiz in den nächsten Jahren tendenziell ihre Ressourcen in einem gemeinsamen Verwaltungssystem integrieren werden, wodurch die Anforderung des Deduplizierens nicht mehr ganz so stark im Mittel-

³³ www.worldcat.org

punkt stehen wird. Das Grundproblem bleibt jedoch bestehen: Es sollen Ressourcen möglichst automatisiert zusammengeführt oder miteinander in Verbindung gestellt werden. Diese Anforderung verlangt in anderen Kontexten auch in Zukunft nach einer Lösung. Eine starke Motivation und Triebfeder für diese Arbeit war, Alternativen und/oder Ergänzungen zu den bestehenden grundsätzlich bewährten Verfahren zu finden. Optimal ist (möglichst) eine Plattform, welche Methoden und Verfahren für folgende Zielsetzungen anbietet:

- Die beschriebenen unterschiedlichen Anforderungen besser vereinbart.
- Die bisherigen sehr weitgehenden Möglichkeiten der bibliothekarischen Datenaufbereitung unseres bisherigen Datenhubs unterstützt.
- Die Fähigkeiten der extrem performanten Verarbeitung heutiger stream processing Systeme anbietet.

3.2.3 Skalierbarkeit einzelner aktueller Komponenten

3.2.3.1 Skalierbarkeit des Datenhubs

Von Beginn an setzen wir beim Datenhub auf die CBS genannte Komponente der Fa. OCLC. Diese wird vor allem bei grösseren Bibliotheksverbünden als zentraler Katalog (interaktiver Nutzerbetrieb) sowie Managementlösung für Metadaten eingesetzt³⁴. In swissbib kennen wir keinen Nutzerdialog, die von uns genutzten Prozesse zur Deduplizierung, Anreicherung und Clustering sind automatisiert und batchbasiert.

Als Storagesystem verwendet CBS die relationale Datenbank Sybase (von der Leistungsfähigkeit mit Oracle zu vergleichen). Meine Annahme: Dies ist ein wesentlicher Punkt, warum die Laufzeit von Batchverarbeitungsprozessen sehr hoch werden kann. Die Initialisierungsphase, bei der Entities aus den aktuell rund 20 verschiedenen Datenquellen geladen und aufbereitet werden (Deduplizierung, Clustering, Anreicherung) benötigt mit der aktuellen Technologie nicht unter 14 Tage.

Nach der Initialisierung wird der Datenhub mit den Ergebnissen der täglichen Arbeit der BibliothekarInnen aus den verschiedenen Quellen jede Nacht aktualisiert. Im Tagesdurchschnitt sind dies zwischen 100.000 und 300.00 Messages. Dafür werden im Durchschnitt zwischen 2 bis 6 Stunden benötigt. Für das momentane Volumen sind diese Kennzahlen ausreichend. Wird das Volumen und die Velocity jedoch erheblich

³⁴ <https://www.oclc.org/de-DE/publications/newsletters/enews/2013/33/de-04.html>

grösser, wie in der Einleitung angenommen, sollte die traditionelle Technik durch Methoden und Verfahren aus dem Bereich der BigData Technologien zumindest ergänzt werden.

Auch die kommerziellen Hersteller von Datenmanagementsystem wie CBS gehen diesen Weg. OCLC hat seinen grossen Datenpool Worldcat auf Hadoop und HBase gestützte Verfahren umgestellt.³⁵ Von anderen Firmen wie beispielsweise ExLibris habe ich noch nichts Vergleichbares gehört. Ich vermute, dass dort aktuell die relationale Technik noch vorherrschend ist.

3.2.3.2 Skalierbarkeit des Data Ingestions

Ebenso diskutabel ist der Aspekt der Skalierbarkeit bei unserer jetzigen Data Ingest Komponente (vgl. Kap. 3.2.1) und zwar spätestens dann, sollten wir uns dazu entscheiden, in Zukunft zum Beispiel auch interaktive Benutzerdaten im Streamingverfahren und Near Real Time zu analysieren. Zwar liesse sich auch hier die bestehende Komponente weiter ausbauen und manuell parallelisieren, sicher würde dabei die Kohäsion jedoch weiter abnehmen und heutige producer / consumer – groups eines modernen Event-Hubs nehmen einem Grossteil dieser Arbeit ab.

4 Lambda / Kappa Architektur als mögliche Basis und erster Schritt hin zu einer zukünftigen swissbib Plattform

Eine zukünftige Lösung sollte:

- die in Kapitel 3 diskutierten Probleme adressieren
- die Notwendigkeit des Big-Bang eines komplett neuen Systems vermeiden und eine evolutionäre Weiterentwicklung ermöglichen, bei der bisher bewährte Verfahren mindestens so lange weiter eingesetzt werden können, bis eine adäquate und den zukünftigen Anforderungen 100% gewachsene Lösung entstanden ist. Ich gehe auch davon aus, dass der bisher eingesetzte traditionelle data hub, mit den in vielen Jahren entwickelten und bewährten Verfahren für das bibliographische data management, seine Qualitäten gut zusammen mit neuen Methoden und Verfahren einbringen kann.

³⁵ <https://www.oclc.org/news/releases/2013/201329dublin.en.html>

Einen ersten Vorschlag für die zukünftige Lösung zeigt Abb. 5

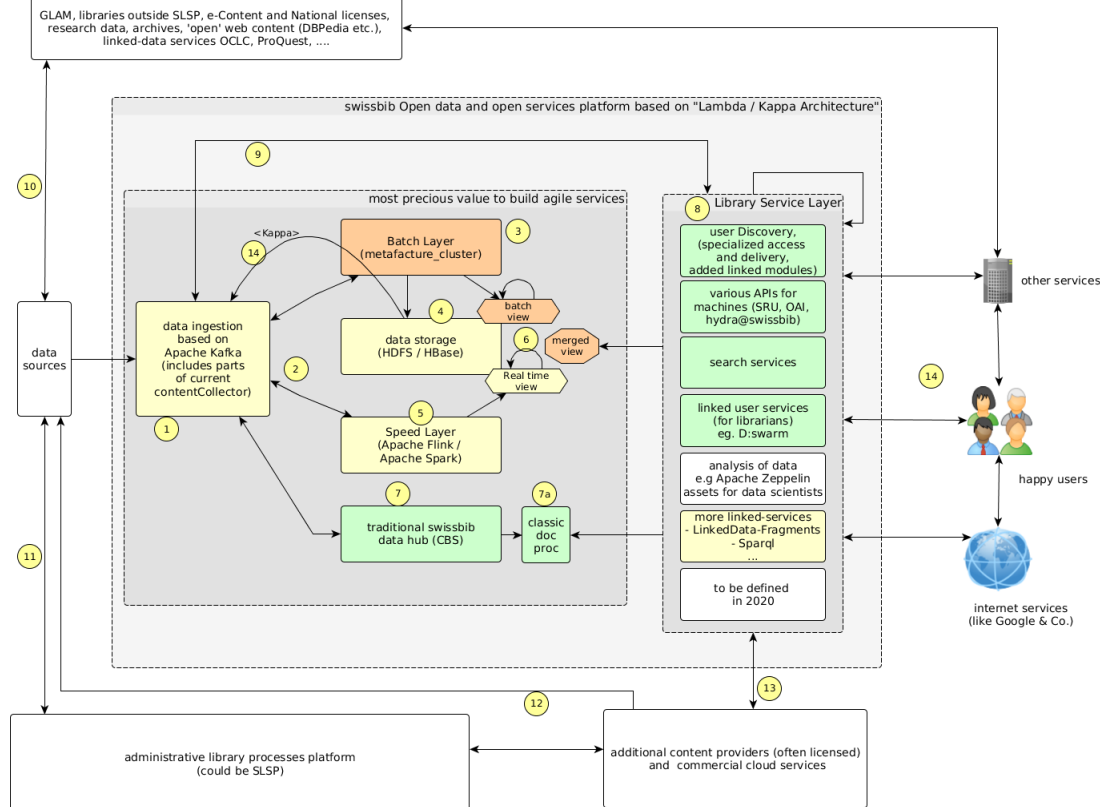


Abb. 5: Infrastruktur auf Basis der Lambda / Kappa Architektur
Quellenhinweis: eigene Abbildung

Beschreibung der Komponenten von Abb. 5:

- **data ingestion (1,2):** Dieses wird in Zukunft auf dem Event-Hub Kafka basieren. Er löst die bisher verwendete Komponente grundsätzlich ab und vereinfacht das Verfahren damit erheblich. Es ist jedoch möglich, einen grösseren Teil der bisherigen Implementierung in Python entweder in der Rolle des Kafka Produzenten oder in der des Kafka Consumers wiederzuverwenden. Ein Consumer wird sicherlich für die Kommunikation zwischen Kafka und unserem bisherigen Data-Hub implementiert werden müssen (Es existiert kein Kafka Adapter für CBS). Aber auch hier kann die Implementierung zum grösseren Teil aus der bisherigen wiederverwendet werden. Im Rahmen der Wiederverwendung wird sich der Code sowohl für Produzenten als auch Konsumenten

erheblich vereinfachen (vgl. die Ausführungen zu Kohäsion im Kap. 3.2.2.1). Das neue Verfahren wird im Rahmen eines Prototyps im Arbeitspaket 2 (s. Kap. 5.1) implementiert.

- Der sog. Batch-Layer (3) der Lambda Architektur, wie er in [3] von Marz und Warren beschrieben wird, kann in der neuen Architektur durch die `metafacture_cluster` Komponente³⁶ der Deutschen Nationalbibliothek umgesetzt werden. Als storage System (4) verwendet `metafacture_cluster` das verteilte Dateisystem HDFS zusammen mit der spalten-orientierten Datenbank HBase. Als Teil dieser Arbeit beginne ich im Rahmen des Arbeitspakets 3 (Kap. 5.2) mit der Evaluierung (und ersten Prototypimplementierung) dieses Konzepts. Damit sollen erste Aussagen gemacht werden können, ob sich ein Batch Layer unter Einsatz von `metafacture_cluster` mit HBase für die Zwecke von swissbib als eine Erweiterung/Alternative zum bestehenden Datenhubs CBS als praktikabel erweisen könnte.
- Batch Views (6) werden in der Lambda Architektur wiederkehrend erstellt. Sie werden dem service layer als Grundlage für seine Dienste entweder einzeln oder als sog „Merged-View“ zusammen mit der „Real-Time-View“ des Speedlayers zur Verfügung gestellt. Auf Seite 15 von [3] wird allgemein beschrieben, wie die Batchview als das Ergebnis einer Funktion auf den kompletten Datenbestand gesehen werden kann und anschliessend als Parameter in die Abfragefunktion eines Dienstes des service layers Eingang findet.

batch view = function (all data)

query = function (batch view)

Übersetzt auf unser reales System bedeutet dies: Der Batch Layer dedupliziert und clustert periodisch (möglichst täglich) unseren ‚kompletten‘ Datenbestand (aktuell knapp 30 Millionen Aufnahmen). Das Ergebnis (die Batch View) wird in unserem heutigen wichtigsten Anwendungsfall komplett mit unserer Suchmaschine SOLR indiziert. Ob das realistisch ist muss noch evaluiert werden. Für unmöglich halte ich dies nicht, wurden doch die cluster der Deutschen Nationalbibliothek mit erheblich mehr Daten (ca. 80 Millionen) bereits im Jahre

³⁶ Vgl. <https://github.com/culturegraph/metafacture-cluster>

2013 innerhalb weniger Stunden erstellt. Dies sind jedoch an dieser Stelle rein theoretische Überlegungen und diese müssen, sollte man sich für einen BatchModus auf Basis von metafactory_cluster entscheiden, noch weiter untersucht werden. Anstatt eine komplette BatchView zu erstellen, wie es i.d.R. beschrieben wird, könnten mit Verfahren auf Basis von Hadoop MapReduce, wie sie metafactory_cluster zur Analyse verwendet, evtl. auch nur die Deltas der täglichen Updates verarbeitet werden. Dies ist die Praxis unseres aktuellen Systems. Die Erfahrungen aus dieser Arbeit haben zum Ziel, hier erste Ansätze zu liefern.

- In der Literatur tritt der Speed-Layer (5) parallel zum Batch Layer in Aktion [siehe z.B. bei 3 Kapitel 6,7,12] Seine Aufgabe besteht in diesem Szenario vor allem darin, Daten, die aufgrund der Latenz zwischen den Batchprozessen temporär nicht für Servicezwecke bereitgestellt werden können, zu verarbeiten und in einer sog. Speed-View bereitzustellen. Batch-View und Speed-View werden in der Theorie zur sogenannten Merged-View zusammengeführt und dienen dem service layer als Grundlage für seine Dienste. Soweit zur Theorie. Für den swissbib Betrieb hat die Anforderung der nahezu Real Time Verarbeitung von bibliographischen Daten zumindest momentan wenig Relevanz. Dies war ein Grund, warum ich in der Planung zu dieser Arbeit Stream Processing Verfahren nicht an erster Stelle in meine Arbeitspakete mit aufgenommen habe, sondern lediglich erste Schritte in Richtung dieses Themas einschlagen wollte. Je mehr ich mich jedoch im Rahmen von Kap. 5.2 in das Thema metafactory_cluster einarbeitete, dort meine Erfahrungen sammelte und mich schliesslich verstärkt in die Thematik Stream Processing mit Apache Flink³⁷ begab, desto stärker gelangte ich zu der Überzeugung, dass vorhandene Hadoop MapReduce Implementierungen, die 2013 innerhalb metafactory_cluster implementiert wurden, aus meiner jetzigen Sicht relativ gut auf das Stream / Batch Verarbeitungsmodell von Apache Flink umgestellt werden könnten. Nähere Details zu diesen Überlegungen im Abschlusskapitel 5.3. Abb.5 enthält den Hinweis auf diese Entwicklung durch den Bezug auf die sog. Kappa – Architektur. Hier werden Daten aus dem Event Hub in einem „Ladeprozess“ in einem dauerhaften storage auf Basis von z.B. HDFS/ HBase persistiert. Das kön-

³⁷ Vgl. <https://flink.apache.org/>

nen initiale Ladeprozesse oder regelmässige Updates sein, wie das gängige Praxis in swissbib ist. Es gibt jedoch kein Pre-Processing mit anschliessender Batch-View mehr, sondern die Daten werden, erneut über Kafka (14), dem stream processing zur Verfügung gestellt. Hier entfällt der Batch-Layer ausser dem storage komplett (weswegen er in Abb. 5 schwach rot markiert worden ist). Ob Kafka als data-source wie im theoretischen Modell genutzt wird oder die stream processing Implementierung den data storage direkt als source verwendet ist für mich im Moment nur zweitrangig. Im Kapitel 5.1 implementiere ich den Ladeprozess in einen HBase storage durch eine Erweiterung von metafactory_cluster. Im Rahmen eines Prototyps binde ich in Kapitel 5.3 sowohl HBase als auch Kafka als source für Flink ein.

- Unser traditioneller Daten-Hub (7) kann in diese Architektur nahtlos integriert werden. Er beliefert unsere bisherigen Services, die neu Teil des Service Layers sind, in der bisherigen Art und Weise (7a) auf Basis des vorhandenen document processing.
- Nicht unwichtig ist darauf hinzuweisen (erkennbar an den bi-direktionalen Pfeilen in (2)), dass vor allem der Speed Layer nicht nur Konsument des Event Hubs ist sondern auch in die Rolle des Produzenten treten kann (Ergebnisse der Prozesse des Layers werden in den Event-Hub erneut eingestellt, so dass sie durch andere Prozesse wiederverwendet werden können. So ist zum Beispiel denkbar, dass der traditionelle Daten-Hub seine Ergebnisse dort ablegt, die dann zum Beispiel im Rahmen des Speed-Layers für eine vertiefte Datenanalyse verwendet werden könnten.
- Die Services der Infrastruktur werden neu im Service-Layer gebündelt. Hier sind die bereits jetzt produktiv eingesetzten (mit stärkerem Grün markiert) natürlich enthalten. Komponenten des Service Layers müssen nicht ausschliesslich auf den Views des Datenlayers basieren, sondern können auch Schnittstellen von Komponenten innerhalb des Services Layers benutzen (discovery → Suchmaschine) (8, 8a) Gleichzeitig ist auch denkbar, dass Daten aus den Topics des Event-Hubs genutzt oder dort eingestellt werden (9). Beispiele könnten Serverlogs sein oder Interaktionsdaten von Benutzern.
- Es ist auch möglich, dass einzelne Komponenten des Service Layers direkt Daten von externen Repositorien benutzen (13). Dies wird bereits heute

gemacht, zum Beispiel mit dem Zugriff auf den lizenzierten Artikel-Content von kommerziellen Anbietern über eine API ³⁸

- Die Möglichkeiten des data ingestion sind nahezu unbegrenzt, nicht zuletzt durch die enorme Skalierbarkeit des Event-Hubs Kafka. So können Daten aus den verschiedensten Quellen aufgenommen (10, 11, 12) und in die Verfahren der Datenanalyse des Batch und Speed Layers und/oder den traditionellen Datenhub überführt werden.
- Weiterentwickelt wird die neue Infrastruktur nur deswegen, damit die NutzerInnen (Mensch und / oder Maschine) den gewünschten Dienst erhalten (14). Diese Wünsche sind natürlich, häufig innerhalb kurzer Zeiträume, sehr volatil. Die aufgezeigte Infrastruktur ermöglicht es jedoch, die von den BenutzerInnen verlangten Services (auch proaktiv) sehr agil bereitzustellen.

5 Evaluation / Implementierung einzelner Komponenten der neuen weiterentwickelten swissbib Plattform im Rahmen von Prototypen

Da ich im Rahmen dieser Projektarbeit ausschliesslich Prototypen implementieren kann und diese vor allem als eine Art „Proof of concept“ bzw. Machbarkeitsstudie dienen sollen, installiere ich die notwendige Infrastruktur nur auf lokaler Hardware.

Es steht mir ein relativ leistungsfähiger Laptop zur Verfügung

model name: Intel(R) Core(TM) i7-4800MQ CPU @ 2.70GHz

cpu core: 4

MemTotal: 24619628 kB

SSD Disk 500 GB

OS: Linux, Ubuntu 16.04

5.1 Apache Kafka als Event Hub

5.1.1 Installation von Kafka

Ich beschränke mich zu Beginn auf eine Defaultinstallation ohne grössere Konfigurationen. Es wird ein Zookeeper plus ein Event Broker benötigt. Ich verzichte zu Beginn

³⁸ Vgl. <http://baselbern.swissbib.ch/Summon/Home>

aus Einfachheitsgründen auch auf ein Zookeeper Ensemble. Ebenso beschränke ich mich auf eine Instanz eines Kafka Brokers. Dies sieht dann wie folgt aus:

```
drwxrwxr-x 4 swissbib swissbib 4096 Aug 11 22:00 ./
drwxrwxr-x 24 swissbib swissbib 4096 Aug 13 19:44 ../
lrwxrwxrwx 1 swissbib swissbib 19 Aug 5 12:58 kafka -> kafka_2.11-0.10.0.0/
drwxr-xr-x 7 swissbib swissbib 4096 Aug 5 12:58 kafka_2.11-0.10.0.0/
lrwxrwxrwx 1 swissbib swissbib 15 Aug 5 12:42 zk -> zookeeper-3.4.8/
drwxr-xr-x 11 swissbib swissbib 4096 Aug 5 12:44 zookeeper-3.4.8/
swissbib@ub-sbhp02:~/environment/tools/dataIngestion$
```

Abb. 6: Installation Kafka / Zookeeper
Quellenangabe: eigene Abbildung

5.1.2 Entwicklung eines Konsumenten für den Kanal OAI

OAI ist der Kanal, der im swissbib Projekt zur Zeit am häufigsten genutzt wird, um Daten für die Plattform bereitzustellen. Dieses Protokoll wird von den grossen kommerziellen Bibliothekssystemen grundsätzlich unterstützt.

Da ein Ziel die Wiederverwendung möglichst grösserer Teile der bestehenden Implementierung ist, können folgende Requirements genannt werden:

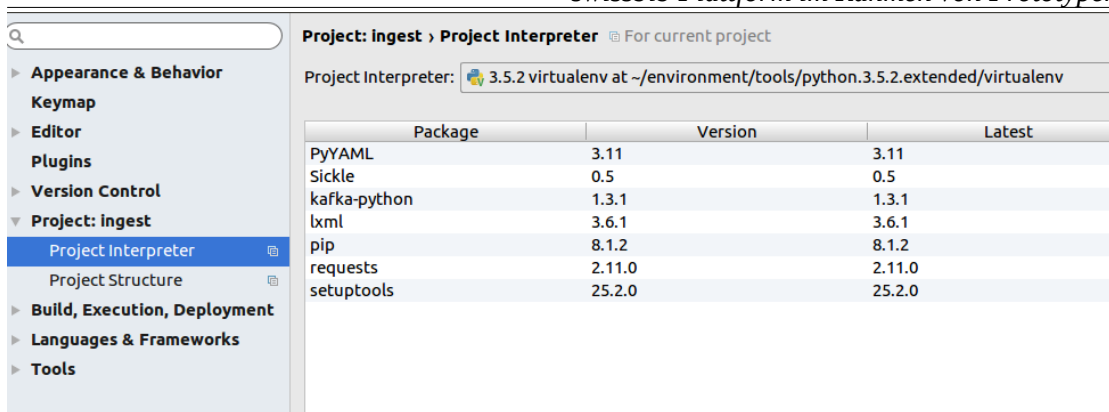
- Python in der Version 3.x als Programmiersprache. Die bisherige Implementierung basiert noch auf der Version 2.7. Der Hauptgrund dafür: Es wird eine (damals sehr populäre, heute gibt es mehr Alternativen) externe Library verwendet, für die es bis jetzt immer noch keinen Upgrade für die Version 3.x gibt.
- Die Weiterentwicklung sollte als Gelegenheit genutzt werden, ‚Altlasten‘ zu beheben. Stichworte: stärkere Modularisierung, einfachere Konfiguration, Erhöhung der Kohäsion wie in Kap. 3.2.2.1 diskutiert.

Das Repository für den ersten Prototyp ist auf Github abrufbar³⁹

Die von mir genutzte Umgebung mit den Abhängigkeiten lässt sich aus der Abb. 7 gut erkennen.

³⁹ <https://github.com/guenterh/dataIngestion>

5. Evaluation / Implementierung einzelner Komponenten der neuen weiterentwickelten swissbib Plattform im Rahmen von Prototypen



Package	Version	Latest
PyYAML	3.11	3.11
Sickle	0.5	0.5
kafka-python	1.3.1	1.3.1
lxml	3.6.1	3.6.1
pip	8.1.2	8.1.2
requests	2.11.0	2.11.0
setuptools	25.2.0	25.2.0

Abb. 7: Abhängigkeiten der Python Umgebung
Quellenangabe: eigene Abbildung

Man sieht, die Abhängigkeiten sind sehr gering. Sie beschränken sich auf den notwendigen Kafka -client sowie eine library zur Unterstützung des OAI Protokolls. Die vor 6 Jahren noch nicht vorhandene Sickel Komponente unterstützt Python 3.x und ist in der Nutzung sehr einfach. Ihr Slogan: „Sickle: OAI-PMH for Humans“⁴⁰ hat sich für mich in der ersten schnellen Nutzung bestätigt. PyYAML unterstützt die Nutzung von YAML Konfigurationsdateien⁴¹. Von den bisher verwendeten XML Konfigurationen möchte ich Abstand gewinnen.

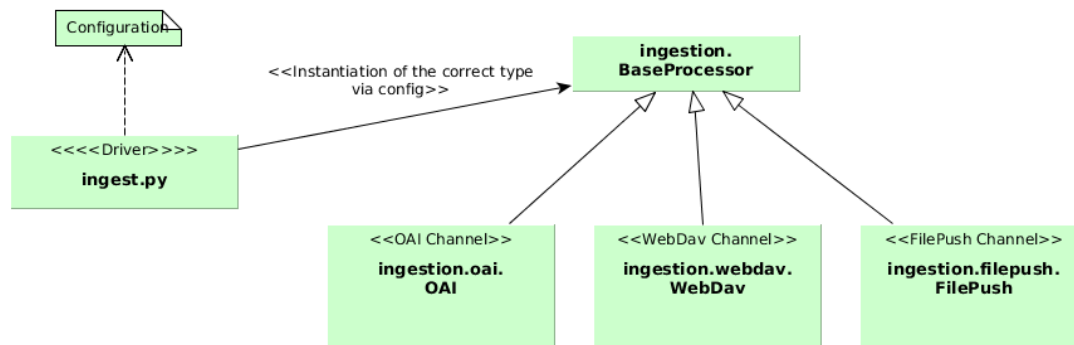


Abb. 8: Klassendiagramm Kafka Produzenten
Quellenangabe: eigene Abbildung

Die Klassenstruktur der Produzenten, wie in Abb. 8 gezeigt, ist sehr übersichtlich und einfach. Ein Treiber instantiiert mit Hilfe der Konfiguration den für den jeweiligen Kanal passenden Konsumenten. Dieser enthält seine notwendigen Informationen (Zeitin-

⁴⁰ Vgl. <https://sickle.readthedocs.io/en/latest/>

⁴¹ Als Beispiel für die data source OAI HSG <https://github.com/guenterh/dataIngestion/blob/master/config/files/idssg1.oai.yaml> In XML sieht dies ein wenig komplizierter aus <https://github.com/swissbib/contentCollector/blob/master/example-Configs/oai.channel.xml> . Allerdings enthält diese mehr Aspekte als der momentane Prototyp in YAML

tervall zur Abfrage der data source sowie Verbindungsdaten zum Kafka Broker ebenfalls aus der Konfiguration. In einem ersten Testlauf, mit dem die kompletten Daten der Hochschulbibliothek St. Gallen (kleinste Hochschulbibliothek) abgeholt wurden, dauerte dieser Vorgang knapp 1.5 Stunden. Diese Zeitdauer basiert vor allem aufgrund der Übertragungslatenz. Kafka selber war während dieser Zeit auf meiner Maschine kaum ausgelastet. Insgesamt wurden etwas mehr als 900.000 messages abgeholt.

```
swissbib@ub-sbhp02:~$KAFKA_RUN/kafka-run-class.sh kafka.tools.GetOffsetShell  
--broker-list localhost:9092 --topic OAI --time -1  
OAI:0:916816
```

5.1.3 Erste Beurteilung des Prototyps und next steps

Die Installation der Kafka Infrastruktur ist recht intuitiv. Meine Vorkenntnisse im Einsatz von SolrCloud bei swissbib, eine Technik die ebenfalls den Koordinationsserver Zookeeper einsetzt, kamen mir zugute. Die unter [1][2][4][5] angegebene Literatur habe ich zumindest überflogen und einzelne Abschnitte oder Videosequenzen etwas vertiefter betrachtet. Dies gibt einem ein ausreichendes Grundverständnis für die Architektur, so dass ich bald damit beginnen konnte mir zu überlegen, wie ein Produzent unter den angegebenen Rahmenbedingungen aussehen sollte. Insgesamt wendete ich für dieses erste Arbeitspaket für den aktuellen Stand ein gutes Wochenende auf.

Für mich haben sich meine Erwartungen, wie in den konzeptionellen Abschnitten beschrieben, erfüllt und ich bin nach wie vor davon überzeugt, dass Kafka ein idealer Nachfolger / Weiterentwicklung unserer bestehenden data ingest Komponente werden kann. Es muss natürlich noch einiges implementiert und optimiert werden. Im Rahmen von Kafka wurde noch überhaupt nichts speziell konfiguriert und sog. Producer-Groups würden die Geschwindigkeit des data ingestions erheblich erhöhen. Allerdings könnten damit die OAI Server der Bibliothekssysteme ein Problem bekommen. Groups sehe ich deshalb eher bei den Kanälen File-Push und WebDav. Diese wurden bisher jedoch noch nicht implementiert. Eine Instantiierung wirft zur Zeit noch eine ‚not implemented‘ exception. Bei diesen beiden Kanälen dürfte das Wiederverwendungspotential unserer produktiven Komponente auch erheblich höher sein, da ich keine neue externe library (Sickle) wie im Prototyp verwenden werde.

5.2 Metafacture Cluster als Komponente eines Batch Layers

5.2.1 Einführung in Metafacture und Metafacture_cluster

Metafacture ist ein Framework zur Transformation und Analyse von Metadaten. Metadaten gibt es nicht nur in der Bibliothekswelt in reichhaltiger Form für die verschiedensten Anwendungszwecke. Warum diese Vielfältigkeit, die es häufig erforderlich macht, das Daten immer wieder transformiert und für bestimmte Anwendungszwecke speziell aufbereitet werden (müssen)? Auch wenn man einige Erscheinungsformen aus der historischen Entwicklung wird begründen können, ist der Hauptgrund wohl, dass Anwendungen ohne Metadaten ihren Zweck i.d.R. nicht erfüllen können und nicht selten die Formate den Anforderungen der Domäne folgend modelliert werden. Betrachtet man nur die auf Suchmaschinenteknik basierenden Information Retrieval Systeme von Bibliotheken, so basieren die Informationen in den dort vorhandenen Suchdokumenten zu einem grossen Teil auf Metadaten und zu einem kleineren Teil aus Volltexten⁴². In der Bibliothekswelt ist hierfür aktuell immer noch das bereits 1966 entwickelte MARC Format dominierend.⁴³ Im technischen Sinn sind Metadaten semistrukturierte Daten. Warum das? Schaut man sich wieder Bibliotheksdaten an, findet man schnell Beispiele dafür. So werden das Geburts- oder Sterbedatum einer Person häufig nicht als exakte Date-Datentypen encodiert, sondern mit Werten wie „ca. 1610“ oder „150?“. Um mit diesen Angaben umgehen zu können, ist in Transformations- und Analyseprozessen die Normalisierung von Daten i.d.R. ein wichtiger Teilschritt. Zwar gibt es offizielle Regelwerke⁴⁴, deren Anwendung bietet aber häufig Interpretationsspielräume, so dass im konkreten Fall häufig doch wieder Besonderheiten zu berücksichtigen sind⁴⁵. Diese Prozesse waren (und sind) teuer, weil regelmässig mehrere Personen in unterschiedlichen Rollen daran beteiligt sind. Die Domänenspezialistin mit dem Format- und Metadaten Know-How sowie Personen mit einem Programmierhintergrund. Diesen Herausforderungen möchte das Metafacture Framework begegnen. Es besteht in der Hauptsache aus folgenden Teilen:

⁴² Vgl. das Schema der Suchmaschine von swissbib.ch
https://github.com/swissbib/searchconf/blob/master/solr/bib/solr6/SOLR_HOME/sb_biblio/conf/schema.xml sowie ein beispielhaftes Suchergebnis: http://search.swissbib.ch/solr/sb-biblio/select?q=%3A*&wt=xml&indent=true

⁴³ Vgl. https://de.wikipedia.org/wiki/Machine-Readable_Cataloging

⁴⁴ Früher z.B. RAK <http://d-nb.info/986402338/34> heute RDA
https://de.wikipedia.org/wiki/Resource_Description_and_Access

⁴⁵ Vgl. dazu auch Ausführungen in [6 - Introduction]

- einer deklarativen und dem Streamingprinzip folgenden Domain Specific Language (DSL) genannt Metamorph. Die DSL benutzt für ihre Definitionen XML.
- einzelnen Modulen, die zusammen mit der Metamorphkomponente zu pipes (workflows) zusammengebaut werden können. Innerhalb dieser pipes finden die gewünschten Transformationen oder Analysen statt.
- Einem einfachen im Wesentlichen auf Key/Values basierenden Datenmodells, mit dem die oben beschriebenen Anforderungen semistrukturierter Daten adressiert werden sollen.

Das Datenmodell von Metamorph ist dabei bewusst so allgemein gehalten, dass es einen Grossteil der in der Praxis vorkommenden sonstigen Formate (wie bspw. JSON, XML, RDF, CSV, Beacon....) abbilden kann⁴⁶. Die zu workflows zusammengestellten Bausteine müssen miteinander kommunizieren können (Teilschritte des Transformationsprozesses an den nächsten step weiterreichen). Dies wird durch vom Framework definierte Schnittstellen erreicht, über die einfache Strings, Triples, Objekte oder Metadatenereignisse ausgetauscht werden. Zum Abschluss dieser sehr knappen Vorstellung sei noch erwähnt, dass workflows wiederum durch eine eigene DSL (Flux) definiert werden können oder die Module des workflows auch im eigenen Applikationscode zusammengestellt werden können. Letztere Methode setzt Hadoop Mapper und Reducer Typen in metafacture_cluster ein.

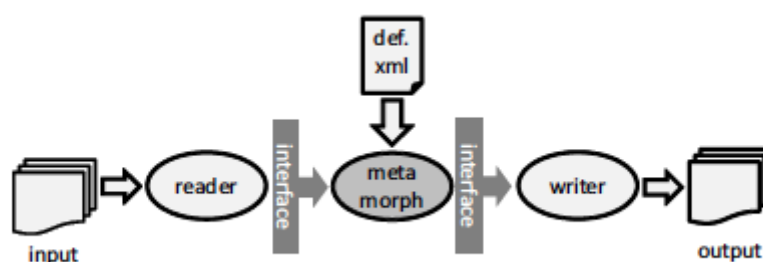


Abb. 9: einfacher workflow bestehend aus reader, Transformation und writer. Die Transformation führt ein Metamorph Modul auf Basis einer übergebenen DSL Definition durch
Quellenhinweis: Geipel, Böhme, Hannemann [6]

Metafacture_cluster benutzt Metafacture um die Transformationen und Analysen auf einem Hadoop Cluster zur Verarbeitung von grossen Datenmengen innerhalb sehr kurzer Zeit durchführen zu können. Metafacture hat die schöne Eigenschaft, dass sich die

⁴⁶ Für weitere Einzelheiten zum Datenmodell vgl. [8]

vorhandenen Module, die zu workflows zusammengestellt werden, erweitern lassen oder neue hinzugefügt werden können. Einzige Voraussetzung: die neuen Module müssen die Schnittstellenbeschreibungen einhalten, da dies die Voraussetzung für die Integration in pipes ist. Wie dieses Prinzip der Erweiterung im spezifischen Fall von metafactory_cluster prinzipiell eingesetzt wird, soll das nachfolgende kleine Klassendiagramm ausdrücken.

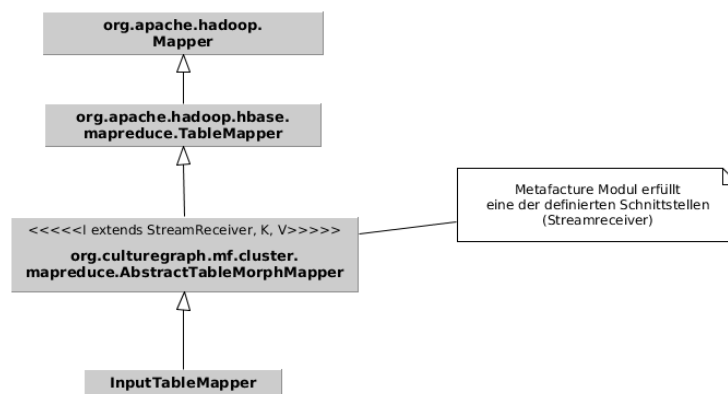


Abb. 10: InputTableMapper als Komponente für MetafactoryWorkflows und Hadoop Mapper mit Anschluss an ein HBase Tabelle für den Bezug von Daten.
Quellenhinweis: eigene Abbildung

Der Typ InputTableMapper kann im Job eines Hadoop MapReduce Prozess als Mapper Typ angegeben werden⁴⁷. Dadurch, dass er von TableMapper aus der HBase library ableitet, erhält der Mapper seine Daten direkt aus der konfigurierten HBase Tabelle geliefert. An dieser Stelle vorab: ich habe diesen use-case im Rahmen der Projektarbeit nicht lauffähig nachvollziehen können. Meine Probleme:

- metafactory_cluster aus dem Jahre 2013 basiert im bestehenden Original noch auf heute veralteten Versionen von Hadoop und HBase. Für meine Zwecke der Projektarbeit habe ich die Versionen angehoben und versucht, die nicht mehr kompatiblen Stellen anzupassen.
- Ich habe einiges an Zeit investieren müssen, um mein Setup, zusammen mit HBase und Hadoop als pseudo-verteilte Instanzen, in den notwendigen Konfi-

⁴⁷ <https://github.com/guenterh/casBigDataBern/blob/master/src/main/java/org/culturegraph/mf/cluster/job/merge/Union.java#L90>

gurationen lauffähig zu bekommen. Im Rahmen des begrenzten Zeitbudgets für die Projektarbeit konnte ich hier nicht noch mehr investieren.

- Nicht nur im Umfeld des Hadoop-Ecosystems hat sich die Software seit 2013 weiterentwickelt, auch bei Metafactory selber (was von metafactory_cluster verwendet wird) gab es Versionssprünge von 1.x zu heute 3.5. So wurde seinerzeit noch eine statische Methode für die Nutzung von Modulen im workflow eingesetzt⁴⁸. Dies alles im Rahmen der Projektarbeit umzustellen war mir nicht möglich.
- Ich benötige mehr Zeit, um die Mechanismen eines MapReduce – jobs in der Komplexität von metafactory_cluster besser zu erfassen um dadurch die noch notwendigen Anpassungen durchführen zu können. Dies war im Rahmen der Projektarbeit mit mehreren Arbeitspaketen und Schwerpunkten nicht möglich.

Auch wenn ich hier im Rahmen dieses Prototyps die für das swissbib Projekt wichtigen Analysen für das Clustering und weitergehende Statistiken noch nicht umsetzen konnte, ist der Erkenntnisgewinn für mich zum jetzigen Zeitpunkt ausreichend.

a) Ich verstehe grundsätzlich das Prinzip des Aufbaus.

b) So wie im Klassendiagramm von Abb. 10 gezeigt, wurde das Design seinerzeit zwar erstellt, ob es auch in allen Facetten entsprechend eingesetzt wurde (Komponente für den Metafactory workflow bei gleichzeitiger Verwendung als Adapter für eine HBase Datenquelle und damit Input für den Mapper) ist für mich fraglich. Ich habe bei der Analyse des Sourcecodes bisher nur den Mapper Aspekt gesehen. Die Nutzung dieses Typs innerhalb eines workflows scheint aussen vor geblieben zu sein. Der Einsatz innerhalb eines workflows ist dann wieder so gelöst, dass jeder zum Mapper aus der Tabelle gesendete Informationseinheit (record) einzeln dem Transformationsprozess übergeben wird⁴⁹. Dieses Prinzip wird in dem von mir implementierten Ladeprozess aus einer Kafka Quelle eingesetzt (vgl. Kap. 5.2.3).

⁴⁸ <https://github.com/guenterh/casBigDataBern/blob/master/src/main/java/org/culturegraph/mf/cluster/job/beacon/FileBeacon.java#L109>

⁴⁹ <https://github.com/guenterh/casBigDataBern/blob/master/src/main/java/org/culturegraph/mf/cluster/mapreduce/AbstractTableMorphMapper.java#L63>

5.2.2 Benutztes Setup (Infrastruktur) für das Arbeitspaket 3

Die von mir für das Arbeitspaket 3 aufgebaute Umgebung ergibt sich aus nachfolgenden Abbildungen.

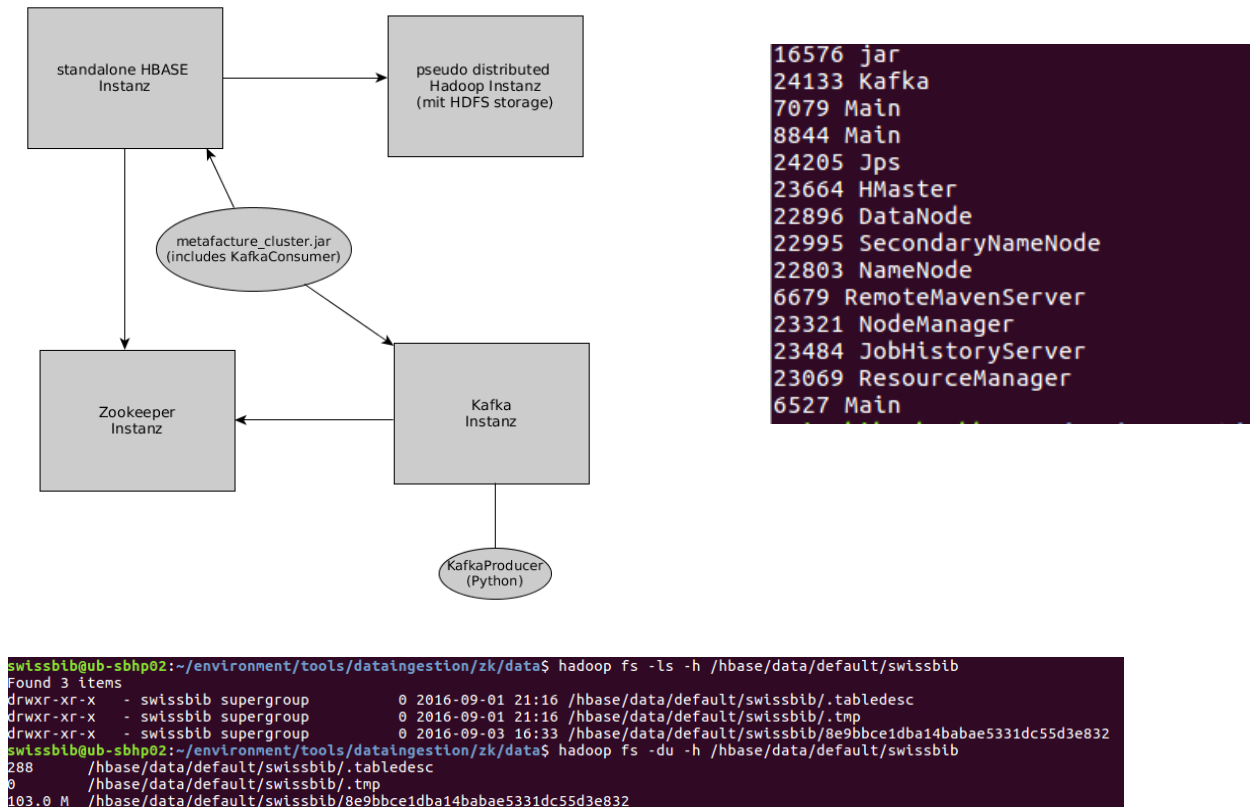


Abb. 11: Setup der Infrastruktur zur Umsetzung von Arbeitspaket 2
Quellenhinweis: eigene Abbildungen

Sämtliche abgebildeten Komponenten laufen auf meinem lokalen Laptop. Die Hadoop Instanzen (NameNode, SecondaryNameNode, DataNode, NodeManager, ResourceManager sowie JobHistoryServer) im sogenannten ‚pseudo-distributed‘ mode. Für diesen Hadoop cluster wurde ein HDFS Dateisystem konfiguriert. Der Hbase server (HMaster) läuft im standalone mode und benutzt das HDFS des Hadoop clusters (dies lässt sich aus dem unteren Screenshot erkennen). Der Kafka-Broker hat seinen eigenen Prozess und benutzt ein Zookeeper-Ensemble (bei mir nur aus einer Instanz bestehend). Letzteres wird auch von HBase verwendet. Das Java-repository metafactory_cluster.jar wurde von mir um einen Kafka-Konsumenten erweitert. Sein Gegenstück (Produzent) aus dem Arbeitspaket 2 ist in Python implementiert.

5.2.3 Laden eines ersten Testdatensets mit Kafka als Datenquelle

Nachdem ich mich wie in Kap. 5.2.1 beschrieben in die grundsätzlichen Zusammenhänge von metafactory_cluster eingearbeitet hatte und ein Zusammenspiel der Infrastrukturkomponenten (Kap. 5.2.2) nach Konsultation von [10], [11], [12] erstmals zustande gekommen war (vor allem die Nutzung des Bündels HBase/Hadoop Cluster durch einen Applikationsclient – bei mir metafactory_cluster – bereitete mir anfangs Mühe weil notwendige Java-Klassenbibliotheken von HBase nicht mit zum Hadoop cluster geschickt wurden), konnte ich mit der Umsetzung des Haupttasks aus dem Arbeitspaket 3, das initiale Laden von Daten unter Nutzung eines Kafka Brokers als Datenquelle, starten. Hierfür musste zuerst ein Konsument geschrieben werden, der die Daten aus einem Kafka Topic liest. Die gelesenen Daten werden dann einem Transformationsprozess übergeben, der das gewünschte spalten-orientierte Datenmodell für HBase erstellt. Letzteres wird, zusammen mit den Rohdaten, in den HBase storage geschrieben. Das Zusammenspiel der Typen ist in Abb. 12 für die beschriebene Aufgabe abgebildet.

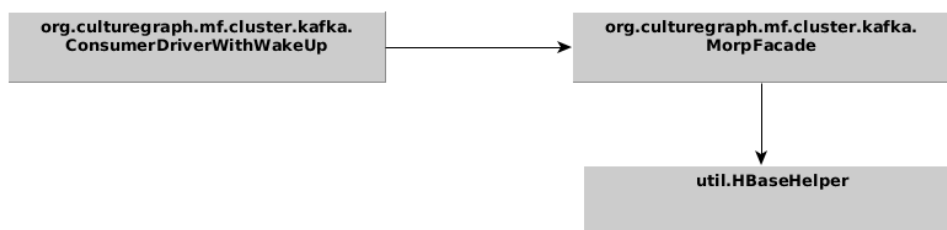


Abb. 12: Komponenten für das Laden von Daten aus Kafka nach HBase
Quellenangabe: eigene Abbildung

ConsumerDriverWithWakeUp⁵⁰ ist so implementiert, dass er Kafka regelmässig anfragt (‚pollt‘), ob neue Daten für das registrierte topic eingestellt wurden. Die MorphFacade ist für die Transformation der Daten in das HBase Schema zuständig. Dazu verwendet sie die bei Start des Consumers angegebene Morpdefinition, die wiederum einem Morphmodul als Teil eines Metafacture workflows übergeben wird⁵¹.

Gestartet wird der Prozess mit folgendem Aufruf:

```
java -cp target/metafacture-cluster-0.0.0-SNAPSHOT-jar-with-  
dependencies.jar:/home/swissbib/environment/tools/dataingestion/kafka/libs/*:. org.-  
culturegraph.mf.cluster.kafka.ConsumerDriverWithWakeUp localhost:9092 swiss-  
bib.oai OAI mapping/ingest.marc21.xml marcxml swissbib
```

Dem Consumertyp wird dabei die Kafka-Broker-Adresse, die GruppenID (jeder Consumer kann für Skalierungszwecke Teil einer Gruppe sein), das Topic, die Datei mit den Morphdefinitionen für die Transformationen in das Datenmodell, das Readerformat sowie der Hbase-Tabellenname angegeben.

An dieser Komponente lässt sich sehr schön die Verkettung von einzelnen Metafacturmodulen über Schnittstellen zu unterschiedlichsten Workflows erläutern. Siehe dazu die beispielhaften Alternativen in Abb. 13

⁵⁰ Die Typen sind unter diesen Adressen abrufbar:
<https://github.com/guenterh/casBigDataBern/blob/master/src/main/java/org/culturegraph/mf/cluster/kafka/ConsumerDriverWithWakeUp.java>
<https://github.com/guenterh/casBigDataBern/blob/master/src/main/java/org/culturegraph/mf/cluster/kafka/MorphFacade.java>
<https://github.com/guenterh/casBigDataBern/blob/master/src/main/java/util/HBaseHelper.java>
Für den Polling Mechanismus habe ich dieses Beispiel <https://github.com/gwenshap/kafka-examples/blob/master/SimpleMovingAvg/src/main/java/com/shapira/examples/newconsumer/simple-movingavg/SimpleMovingAvgNewConsumer.java> als Ideenspender benutzt.

⁵¹
<https://github.com/guenterh/casBigDataBern/blob/master/src/main/java/org/culturegraph/mf/cluster/kafka/MorphFacade.java#L65>

```
//workflow 1
reader = new MultiFormatReader(usedFormat);           //1
Metamorph metamorph = new Metamorph(morpdefinition);  //2
ComplexPutWriter collector = new ComplexPutWriter();   //3
metamorph.setReceiver(collector);                      //4
reader.setReceiver(metamorph);                         //5
reader.read("record from Kafka");                     //6

//workflow 2
Reader reader = new MultiFormatReader(usedFormat);     //7
FormetaEncoder fe = new FormetaEncoder();              //8
fe.setStyle(FormatterStyle.MULTILINE);                 //9
ObjectWriter<String> ow = new ObjectWriter<>("stdout"); //10
fe.setReceiver(ow);                                    //11
metamorph = new Metamorph(morpdefinition);             //12
metamorph.setReceiver(fe);                             //12
reader.setReceiver(metamorph);                         //13
reader.read("record from Kafka");
```

Abb. 13: Illustration wie Metafactory-Workflows zusammengebaut werden können
Quellenangabe: eigene Abbildung

Workflow 1 wird von meinem Kafka-Consumer so verwendet. Der sogenannte Multi-formatReader (ein Repository in dem verschiedene Formatdecoder vorgehalten werden und das mit dem gewünschten Decoder, welcher beim Start des Consumers über die Kommandozeile angegeben wird, initialisiert wird [1]) erhält als Empfänger ein Metamorphmodul zur Transformation der Daten angegeben [5]. Das Metamorphmodul arbeitet die Definitionen der beim Start übergebenen Transformationsdefinition (Morph-DSL) ab [2] und sendet seine Daten an einen sogenannten ComplexPutWriter [4]. Dieser Collector kann die gewünschten Informationen einfach als HBase Put-Typen sammeln und in Bündeln in den Storage schreiben. Der Reader erhält seine Daten von Kafka geliefert [6]

Wie sieht denn nun das Ergebnis einer Metamorphtransformation aus? Dies kann man sich anschauen, wenn man die pipe mit anderen Modulen verkettet, wie das in Workflow 2 exemplarisch gezeigt ist. Der Beginn gleicht der pipe wie in Workflow 1 [7, 13]. Das Metamorphmodul sendet seine Daten nun nicht mehr an einen ComplexPutWriter sondern an einen sogenannten FormetaEncoder [12]. Dieser macht nichts anderes, als das neu erstellte Datenmodell in ein von Menschen besser lesbares Format zu encodieren. Das leicht lesbare Format wird über einen sog. ObjectWriter für Debug-

ging-Zwecke ‚stdout‘ weitergereicht [11]. Auch für diese pipe ist Kafka die Data-Source Quelle [13]. Die Formeta Struktur sieht dann aus wie in Abb. 14 gezeigt.

```
1472928528855 -- waiting for data...
'' {
  'cg:type': 'bib',
  'cg:material': 'book',
  'cg:level': 'single',
  '_id': '000302826',
  'cg:srcId': '000302826',
  'cg:form': ' ',
  'dcterms:issued': '2011',
  'bibo:isbn13': '9783640842537',
  'cg:creator' {
    'name': 'Steiger, Andrea',
    'rel': 'und',
    'cnt': '1'
  },
  'dcterms:title': 'Motivation - Autonomie schaffen, Selbstwirksamkeit erhöhen, Attributionen verbessern',
  'cg:pubName': 'GRIN',
  'dcterms:extent': '40 S.'
}
```

Abb. 14: Ergebnis einer Metamorphtransformation im Formeta-Format
Quellenangabe: eigene Abbildung

Man kann sich bereits jetzt vorstellen, wie aus diesen einzelnen Elementen (plus dem kompletten Rohdatensatz von Kafka, der dem HBase Put Typ als zusätzliche column mitgegeben wird ⁵²) im ComplexPutWriter das Modell (die Columns) für die HBase Datenbank erstellt wird. Die Columns werden als Input benötigt, um in späteren Analyseverfahren zum Beispiel das für swissbib wichtige Clustering und eine Deduplizierung von Records durchzuführen. Domänenexperten (häufig auch Metadaten spezialistInnen genannt) sind über Anpassungen der Metamorphscripte in der Lage, die Eingangselemente für die Algorithmen leicht anzupassen. Das in unserem Fall verwendete Metamorphscript ist hier⁵³ zu finden. Auf die Details der DSL möchte ich im Rahmen dieser Projektarbeit nicht eingehen. Bei Interesse lassen sich unter [8] und [9] weitere Informationen finden.

⁵² <https://github.com/guenterh/casBigDataBern/blob/master/src/main/java/org/culturegraph/mf/cluster/kafka/MorphFacade.java#L84>

⁵³ <https://github.com/guenterh/casBigDataBern/blob/master/src/main/resources/mapping/ingest.-marc21.xml>

5.2.4 Erläuterung des entstandenen spalten-orientierten Datenmodells unter Verwendung von Metamorph

Das Modell der von uns zum Laden verwendeten HBase Tabelle ergibt sich bereits aus den Ausführungen des vorhergehenden Abschnitts. Die nachfolgende Abbildung soll das nochmals verdeutlichen.

```
000860803 mp=1473191155129, value=
000860803 column=prop:x1E<cg:subj>x1E-name\x1FABwelchendes Verhalten\x1E>\x1E, timestamp=1473191155129, value=
000860803 column=prop:x1E<cg:subj>x1E-name\x1FHochschulschrift\x1E>\x1E, timestamp=1473191155129, value=
000860803 column=prop:x1E<cg:subj>x1E-name\x1F\xC3\xA4dophille\x1E>\x1E, timestamp=1473191155129, value=
000860803 column=prop:x1E<cg:subj>x1E-name\x1FResozialisierung\x1E>\x1E, timestamp=1473191155129, value=
000860803 column=prop:x1E<cg:subj>x1E-name\x1FSoziale Arbeit\x1E>\x1E, timestamp=1473191155129, value=
000860803 column=prop:x1E<cg:subj>x1E-name\x1FStrafvollzug\x1E>\x1E, timestamp=1473191155129, value=
000860803 column=prop:cg:form\x1F, timestamp=1473191155129, value=
000860803 column=prop:cg:level\x1Fsingle, timestamp=1473191155129, value=
000860803 column=prop:cg:material\x1Fbook, timestamp=1473191155129, value=
000860803 column=prop:cg:srcid\x1F000860803, timestamp=1473191155129, value=
000860803 column=prop:cg:type\x1FBib, timestamp=1473191155129, value=
000860803 column=prop:dcterms:extent\x1F58 Seiten, timestamp=1473191155129, value=
000860803 column=prop:dcterms:issued\x1F2015, timestamp=1473191155129, value=
000860803 column=prop:dcterms:title\x1FResozialisierung p\xC3\xA4dophiler Straft\xC3\xA4ter in Straf- und Massnahmenvollzug, timesta
mp=1473191155129, value=
000860803 column=prop:raw, timestamp=1473191155129, value=<record xmlns:marc="http://www.loc.gov/MARC21/slin" xmlns:xsi="http://
www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://www.loc.gov/standards/marcxml/schema/MARC21slin.xsd"><marc
:leader> nan a22 4c 4500</marc:leader><marc:controlfield tag="FMT">BK</marc:controlfield><marc:controlfield tag="L
DR"> nan a22 4c 4500</marc:controlfield><marc:controlfield tag="001">000860803</marc:controlfield><marc:controlfie
ld tag="008">100620s2015 sz m 00 ger d</marc:controlfield><marc:datafield tag="040" ind1=" " ind2=" "><marc
:subfield code="a">S2ZUIDS FHS</marc:subfield><marc:subfield code="b">ger</marc:subfield><marc:subfield code="e">rd</marc
:subfield></marc:datafield><marc:datafield tag="100" ind1="1" ind2=" "><marc:subfield code="a">M\xC3\xB4ller, Ornella</mar
c:subfield><marc:subfield code="e">Verfasser</marc:subfield><marc:subfield code="4">aut</marc:subfield></marc:datafield><marc:
datafield tag="245" ind1="1" ind2="0"><marc:subfield code="a">Resozialisierung p\xC3\xA4dophiler Straft\xC3\xA4ter in
Straf- und Massnahmenvollzug</marc:subfield><marc:subfield code="c">Bachelorarbeit von Ornella M\xC3\xB4ller</marc:subfiel
d></marc:datafield><marc:datafield tag="264" ind1=" " ind2="1"><marc:subfield code="a">St.Gallen</marc:subfield><marc:subfiel
d code="b">FHS St.Gallen, Studienrichtung Sozialarbeit</marc:subfield><marc:subfield code="c">2015</marc:subfield></marc:
datafield><marc:datafield tag="300" ind1=" " ind2=" "><marc:subfield code="a">58 Seiten</marc:subfield><marc:subfield co
de="b">Illustrationen</marc:subfield></marc:datafield><marc:datafield tag="336" ind1=" " ind2=" "><marc:subfield code="a">
Text</marc:subfield><marc:subfield code="b">txt</marc:subfield><marc:subfield code="2">rd</marc:subfield></marc:datafield><marc:da
tafield><marc:datafield tag="337" ind1=" " ind2=" "><marc:subfield code="a">ohne Hilfsmittel zu benutzen</marc:subfield><marc:
subfield code="b">nc</marc:subfield><marc:subfield code="2">rd</marc:subfield></marc:datafield><marc:datafield tag="338" ind1=" " ind2=" "><marc:subfield code="a">Band</marc:subfield><marc:subfield code="b">nc</marc:subfield><marc:subfiel
d code="2">rd</marc:subfield></marc:datafield><marc:datafield tag="490" ind1="0" ind2=" "><marc:subfield code="a">Bachelor of Science FHO in Sozialer Arbeit, Studienrichtung Sozialarbeit. Bachelor Thesis</marc:subfield><marc:subfield
code="l">S13</marc:subfield><marc:subfield code="v">S13</marc:subfield><marc:subfield code="w">000775140</marc:subfield></marc:datafield><marc:datafield tag="520" ind1=" " ind2=" "><marc:subfield code="b">Die Arbeit beschreibt die Resozialisie
rung p\xC3\xA4dophiler Straft\xC3\xA4ter in Straf- und Massnahmenvollzug. Vor den Hintergrund des Risikoorientierten Sankt
ionenvollzugs (ROS) beleuchtet sie das Spannungsfeld Sozialer Arbeit zwischen R\xC3\xB4ckf\xC3\xA4hrung und soziale
r Reintegration. (Quelle: Abstract)</marc:subfield></marc:datafield><marc:datafield tag="650" ind1="0" ind2="7"><marc:subfiel
d code="a">Abwelchendes Verhalten</marc:subfield><marc:subfield code="1">(DE-588)4000320-6</marc:subfield><marc:subfiel
d code="2">gnd</marc:subfield></marc:datafield><marc:datafield tag="650" ind1=" " ind2="7"><marc:subfield code="a">P\xC3\x
A4dophilie</marc:subfield><marc:subfield code="1">(DE-588)4126403-4</marc:subfield><marc:subfield code="2">gnd</marc:subfiel
d></marc:datafield><marc:datafield tag="650" ind1=" " ind2="7"><marc:subfield code="a">Resozialisierung</marc:subfield><marc:
subfield code="1">(DE-588)4128233-4</marc:subfield><marc:subfield code="2">gnd</marc:subfield></marc:datafield><marc:
datafield tag="650" ind1=" " ind2="7"><marc:subfield code="a">Strafvollzug</marc:subfield><marc:subfield code="1">(DE-588)
4057808-2</marc:subfield><marc:subfield code="2">gnd</marc:subfield></marc:datafield><marc:datafield tag="650" ind1=" " in
d2="7"><marc:subfield code="a">Soziale Arbeit</marc:subfield><marc:subfield code="1">(DE-588)4055676-1</marc:subfield><marc:
subfield code="2">gnd</marc:subfield></marc:datafield><marc:datafield tag="655" ind1=" " ind2="7"><marc:subfield code="a">
Hochschulschrift</marc:subfield><marc:subfield code="2">gnd-content</marc:subfield></marc:datafield><marc:datafield tag="
949" ind1=" " ind2=" "><marc:subfield code="b">HFHS</marc:subfield><marc:subfield code="c">B18</marc:subfield><marc:subfiel
d code="j">FB-SA-BA-SA-513</marc:subfield><marc:subfield code="p">HH033514</marc:subfield><marc:subfield code="q">000860
803</marc:subfield><marc:subfield code="r">0000010</marc:subfield><marc:subfield code="9">FHS</marc:subfield><marc:subfiel
d code="1">Bibliothek</marc:subfield><marc:subfield code="3">STUDA</marc:subfield><marc:subfield code="4">03</marc:subfield>
</marc:subfield><marc:subfield code="6">IP</marc:subfield><marc:subfield code="7">In Bearbeitung</marc:subfield></marc:ir
ecord>
```

Abb. 15:Auszug eines rows aus der HBase Ladetabelle
Quellenangabe: eigene Abbildung

Die Spalten entsprechen der Struktur, wie wir sie in der leichter lesbaren Formetastruktur in Abb. 14 gesehen haben. Der komplette Rohdatensatz wurde den Metamorphtransformationen hinzugefügt. Ob sich dies für spätere Analyse- und Statistikverfahren als hilfreich erweisen kann sei hier noch offen gelassen. Mir ging es in diesem Prototyp primär darum, wie eine solche Datenstruktur für HBase als spalten-orientierte und verteilte Hadoop Datenbank im Zusammenspiel mit den Mechanismen des Metafacture-Frameworks aufbereitet werden kann.

Die Beschreibung der kompletten Tabelle lässt sich in der HBase Shell wie folgt abfragen:

```
hbase(main):003:0> describe 'swissbib'
Table swissbib is ENABLED
swissbib COLUMN FAMILIES DESCRIPTION
{NAME => 'prop', BLOOMFILTER => 'ROW', VERSIONS => '1', IN_MEMORY => 'false',
KEEP_DELETED_CELLS => 'FALSE', DATA_BLOCK_ENCODING => 'NONE', TTL => 'FOREVER', COM-
PRESSION => 'NONE', MIN_VERSIONS => '0', BLOCKCACHE => 'true', BLOCKSIZE => '65536', REPLICAS-
TION_SCOPE => '0'}
1 row(s) in 0.0160 seconds
hbase(main):004:0>
```

Es gibt eine column-family (prop) mit der alle columns gruppiert werden. Diese columns werden innerhalb des Metafactory-ComplexPutWriter mit Hilfe eines HBase Put clients der family angefügt. Schaut man sich die Struktur am Beispieldatensatz aus Abb. 15 an, erkennt man, dass die values der ursprünglichen bibliographischen Beschreibung den key und damit den Namen der column bilden. Erwartet hätte ich, dass der value der Beschreibung auch value des keys im Datenmodell geworden wäre. Im Modell (Auszug):

```
prop:cg:level\x1Fsingle, timestamp=xxx, value=
prop:\x1E<cg:subj\x1E-name\x1FStrafvollzug\x1E\x1E, timestamp=xxx,value=
prop:\x1E<cg:subj\x1E-name\x1FResozialisierung\x1E\x1E, timestamp=xxx,value=
```

meine erste ursprüngliche Erwartungshaltung:

```
prop:cg:level, timestamp=xxx, value=single
prop:cg:subj-name, timestamp=xxx,value=Strafvollzug
prop:cg:subj-name, timestamp=xxx,value=Resozialisierung
```

Mit dieser ursprünglichen Idee hatte ich, wie gesagt, den kompletten Wert der bibliographischen Beschreibung angefügt:

```
prop:raw, timespamp, value=[die Beschreibung]
```

Denkt man darüber nach, warum das seinerzeit auf diese Weise implementiert worden ist, gibt es für mich im Moment nur einen Grund: Man wollte die Besonderheit von bibliographischen Beschreibungen abbilden, dass Feldnamen mehrfach benutzt werden

können. Um dies in dem spaltenorientierten Modell von HBase nachzuvollziehen, wurde die Möglichkeit verwendet, dass innerhalb einer gruppierenden Family Millionen von columns angefügt werden können, was häufig für Zeitreihen verwendet wird. Ohne mich mit dem Thema eingehender beschäftigt zu haben, werden bei Zeitreihen jedoch in der Regel unterschiedliche timestamps einer column mit gleichem Namen zugefügt. Dies ist für unseren Anwendungsfall aber kaum zweckdienlich.

5.2.5 Erste Beurteilung des Prototyps und next steps

Ich war in der Lage, die in der Projektskizze⁵⁴ von mir vorgesehenen tasks bis auf den letzten „Durchspielen einzelner workflows. Inwieweit ist das in der Dokumentation beschriebene Clustering auf Basis von Metafactory_cluster möglich?“ abzuschliessen. Gründe für das Auslassen habe ich bereits in Kap. 5.2.1 genannt. Ansonsten wurden die gestellten Erwartungen für mich grundsätzlich erfüllt. Der Ladevorgang in einen verteilten Hadoop – storage unter Einbezug von Kafka war möglich. Die Transformations- und Schreibgeschwindigkeit war hoch, auch wenn ich die Prozesse auf meinem lokalen Laptop laufen lasse. Obwohl mein Testdatenset mit rund 150.000 Datensätzen verhältnismässig klein war, gehe ich davon aus, dass sich dieses Verhalten auf vielfach grössere Volumina im Rahmen eines verteilten storage, wie in HBase zusammen mit HDFS anbietet, skalieren lässt. Der jetzt erstellte storage kann die Grundlage für Prozesse auf dem Batch-Layer einer Lambda Architektur sein.

An dieser Stelle möchte ich eine Schemaübersicht der Workflows, wie sie zum Aufbau des heute noch abrufbaren Culturegraph-Portals⁵⁵ im Jahre 2013 entwickelt wurden, kurz vorstellen, da das Arbeitspaket in diesem Kapitel ja gerade zum Ziel hat, die Möglichkeiten der Datenanalyse (vor allem Deduplizierung, Clustering) auf Basis von metafactory_cluster zu evaluieren.

⁵⁴ <https://github.com/guenterh/casBigDataBern/blob/master/bern/projektskizze.cas.bigdata.2016.guenter.hipler.final.pdf> Die Skizze diente als Grundlage dieser Arbeit und wurde am 23.6.2016 in Bern vorgestellt.

⁵⁵ Dieses ist heute noch unter <http://hub.culturegraph.org/> abrufbar.

auch wenn es in den Details doch einige Unterschiede gibt. Das Culturegraph-Portal setzt Hadoop MapReduce Jobs ein (violett gekennzeichnete Prozesse). Möglichkeiten, die ich erst im Rahmen dieser Projektarbeit versucht habe zu evaluieren. Die blau gekennzeichneten Prozesse findet man ähnlich in unserem linked-swissbib Projekt (s. Nr. 10 der Abb. 2), wo wir mit den Mitteln von Metafacture (ohne MapReduce) workflows u.a. zur Transformation der klassischen bibliographischen Daten nach RDF definieren. Der Unterschied hier: Culturegraph verwendet zur Verlinkung mit externen Daten Metafacture „Bordmittel“⁵⁷ während wir uns im Projekt auf von der Gesis entwickelte Verfahren stützen⁵⁸ (s. auch Nr. 11 der Abb. 2).

Bereits mein erster Lösungsvorschlag für eine zukünftige Architektur der swissbib Plattform in Kap. 4 vereinfacht die workflows im Vergleich mit denen der Abb. 16 durch die Einführung eines Event-Hubs bereits sehr. Culturegraph selber lädt seine Daten noch aus verschiedenen files, wodurch ein zusätzlicher job (job_egIngest.sh) entsteht. Der Lambda Vorschlag aus Kap. 4 hat jedoch das grundsätzliche Problem, dass für den Batch workflow (auf Basis Hadoop MapReduce) als auch für den Streaming workflow unterschiedliche Implementierungen geschrieben werden müssen. Nicht gerade etwas, was zur Komplexitätsreduktion beiträgt.

Die Lösung hierfür kann zur Zeit für mich nur die Verwendung von sog. Stream Prozessoren sein, die sich, wie bereits von mir im Kap. 2.2 erwähnt, auf der Schwelle zum Massenprodukt befinden. Vorteile die ich zur Zeit erkenne:

- Integration des relativ starren MapReduce Paradigma mit eher klassischen iterativen Paradigmen in einem Programmiermodell
- klare, relativ schnell überschaubare Schnittstellen die in verschiedenen Programmiersprachen (auch Scripting mit Python) eingesetzt werden können
- Unterstützung sowohl von Streaming- als auch Batchverfahren in einem Modell.
- Wachsende Zahl an Konnektoren, mit denen unterschiedliche Datenquellen und -sourcen an einen workflow eingebunden werden können.
- Möglichkeit, ein vorhandenes domänenspezifisches Framework wie Metafacture leicht zu integrieren.

⁵⁷ Dabei handelt es sich um die Möglichkeit zum Verketteten von flows (und damit von Datenquellen) im Rahmen von sog. „wormholes“ Vgl. dazu die Informationen auf Seite 31 unter http://swib.org/swib13/slides/boehme_swib13_131.pdf

⁵⁸ Vgl. <https://github.com/linked-swissbib/reshaperdf>

- Einfacheres Deployment der Komponenten und bessere Entwicklungsunterstützung (so meine Erfahrung in diesem Projekt)
- Einbindung weiterer Bibliotheken wie Graphprocessing oder Machine Learning und Konnektoren zu End-User Tools wie Apache Zeppelin, so dass nicht nur Infrastruktur im Hintergrund erstellt wird ohne auf den ersten Blick einen direkten Benutzermehrwert zu erzielen.
- Angebot an sogenannten Table API mit starkem Bezug zum SQL Dialekt, wodurch die zahlreichen BenutzerInnen aus diesem Umfeld einen schnelleren Einstieg in die Anwendung finden. Im sonstigen Apache Hadoop Ecosystem müssen dafür zusätzliche Komponenten wie Hive⁵⁹ genutzt werden

5.3 Erste Evaluation von Streamprozessoren für ein schlankes und modernes data processing auf der swissbib Plattform

Wohl nicht nur nach meiner persönlichen Wahrnehmung ist die Entwicklung von Streamprozessoren vor allem in den letzten zwei bis drei Jahren in sehr grosser Bewegung. Es gibt eine Vielzahl von Projekten und Lösungen. Die zwei, welche sich zur Zeit am besten durchgesetzt und meisten beachtet werden, sind die Apache Projekte Spark und Flink. Beide sind seit 2014 Top Level Projekte der Apache Foundation. Spark genießt recht starke Unterstützung vor allem von grösseren amerikanischen Firmen wie IBM oder Cloudera, Flink hat seine Wurzeln in einem europäischen Forschungsprojekt und wird heute eher als das zukunftsorientiertere Projekt gesehen, da es mit dem Ziel der Unterstützung von Streamverfahren entwickelt wurde und den Batchmode nur als einen Spezialfall des Streaming ansieht⁶⁰. Bei Spark ist das genau umgekehrt, weswegen es grössere Schwierigkeiten hat, in heute vor allem aktuellen Streamingverfahren eingesetzt zu werden.

Ich möchte mich an dieser Stelle gar nicht an solchen Diskussionen beteiligen. Unter anderem weil der Streamingmodus zur Zeit zumindest im Bibliotheksumfeld nicht unbedingt im Fokus steht. Was für mich bei der Auswahl für eine erste Evaluation im Rahmen der Definition einer zukunftsgerichteten Informationsplattform den Ausschlag gegeben hat, ist die möglichst schnelle Einarbeitung in die API der Software. Beide Lösungen bieten APIs für Java, Scala sowie Python an und sind intern überwiegend in

⁵⁹ Vgl <https://hive.apache.org/>

⁶⁰ Zu den Ursprüngen der beiden Projekte vgl. auch https://en.wikipedia.org/wiki/Apache_Flink und https://en.wikipedia.org/wiki/Apache_Spark

Scala implementiert. Flink legt bei der Dokumentation der Schnittstellen seinen Schwerpunkt jedoch auf die Java Benutzergruppe, vernachlässigt sie in meinen Augen zumindest weniger. Dies macht es diesem im Vergleich zur Scala-Gemeinde grossen Personenkreis (zu denen ich auch zähle) sehr viel einfacher, einen Einstieg zu finden. Für mich aktuell das Hauptargument, Flink für die Evaluation im Rahmen dieses Projekts zu wählen. In unserem linked-swissbib Projekt haben wir hingegen bereits erste Erfahrungen mit Spark gesammelt. Mein Kollege⁶¹ setzt es ein, um sog. bibliographische work-cluster zu bilden. Ausgangspunkt für diese Entscheidung war ein Memoryproblem, auf das wir in der Benutzung von Metafactory gestossen waren⁶² und das wir in der für uns notwendigen Kürze so nicht lösen konnten. Ziel ist es, einen work-cluster auf Basis einer ID im RDF Datenmodell zu erstellen. Die ID wird bereits in unserem traditionellen Datenhub generiert. Dafür müssen jedoch die Daten unseres gesamten Bestands auf Basis dieser ID aggregiert und die gebildeten Aggregate im Elasticsearch-Server mit den gewünschten Elementen als RDF persistiert werden.

```
{
  "_index": "testsb_160426",
  "_type": "work",
  "_id": "11004889X",
  "_score": 1.0,
  "_source": {
    "@type": "http://bibframe.org/vocab/Work",
    "@context": "http://data.swissbib.ch/work/context.jsonld",
    "dct:contributor": [
      "http://data.swissbib.ch/person/5973e27-c6c1-322d-b95c-aedc18c278d",
      "http://data.swissbib.ch/person/5973e27-c6c1-322d-b95c-aedc18c278d"
    ],
    "rdfs:label": "http://data.swissbib.ch/work/11004889X",
    "rdf:type": "http://data.swissbib.ch/resource/11004889X",
    "dct:title": [
      "http://data.swissbib.ch/resource/11004889X"
    ],
    "dct:title": [
      "Kitāb Sibawayh : [bi-hāmiṣihī:] Taqrīrāt wa-zubad min ṣarḥ Abi Saʿīd as-Sirāfi, fa-huwa al-kitāb al-wāfir al-wāfi. [bi-asfal as-sahifa: Tahsil ʿayn al-qāḥab min maʿdin ḡawhar al-adab fi ʿilm muḡāzāt al-ʿarab",
      "Kitāb Sibawayh : [bi-hāmiṣihī:] Taqrīrāt wa-zubad min ṣarḥ Abi Saʿīd as-Sirāfi, fa-huwa al-kitāb al-wāfir al-wāfi. [bi-asfal as-sahifa: Tahsil ʿayn al-qāḥab min maʿdin ḡawhar al-adab fi ʿilm muḡāzāt al-ʿarab / Yusuf Ibn Sulaymān Ibn ʿIsā as-Santamari]"
    ]
  }
}
```

Abb. 17: work-cluster im ElasticsearchIndex nach Aggregation durch Spark
Quellenangabe: eigene Abbildung

Solche Aggregate helfen BenutzerInnen im Rahmen von Präsentationskomponenten Suchresultate besser zu überblicken⁶³. Die Lösung findet sich auf GitHub⁶⁴, sie ist sehr übersichtlich und kurz. In einem workflow wird Elasticsearch sowohl als Datenquelle als auch -ziel definiert, wodurch der komplette Bestand (mehr als 20 Millionen Entities) gelesen, im Rahmen einer Map-Funktion bewertet, nach bestimmten Kriterien ein Bündel gebildet und anschliessend zurück in den Index geschrieben wird. Auf einem stand-alone cluster dauert dieser Vorgang wenige Minuten. Auch in der Bibliothekswelt finden die Werkzeuge und Verfahren, wie sie in dieser Projektarbeit evaluiert wer-

⁶¹ Siehe <https://github.com/sschuepbach>

⁶² Vgl. <https://github.com/culturegraph/metafactory-documentation/issues/3>

⁶³ Vgl. <https://www.swissbib.ch/Search/Results?lookfor=11004889X&type=FRBR>

⁶⁴ <https://github.com/linked-swissbib/workConceptGenerator/blob/master/src/main/scala-2.11/org/swissbib/linked/Application.scala>

den, so langsam Einzug. Europeana unterstützt ein sog. „Metadata Quality Assurance Framework“. Hier wird unter anderem Spark eingesetzt.⁶⁵

Nach dieser ersten Einleitung, wie Streaming Prozessoren auch in der Bibliothekswelt eingesetzt werden können bzw. bereits werden, zurück zu meinem eigentlichen Ziel: Suche nach einem schlankeren Modell verglichen mit dem in Abb. 5 aufgezeigten.

Wie dieses in der Schemaübersicht aussehen könnte zeigt Abb. 18

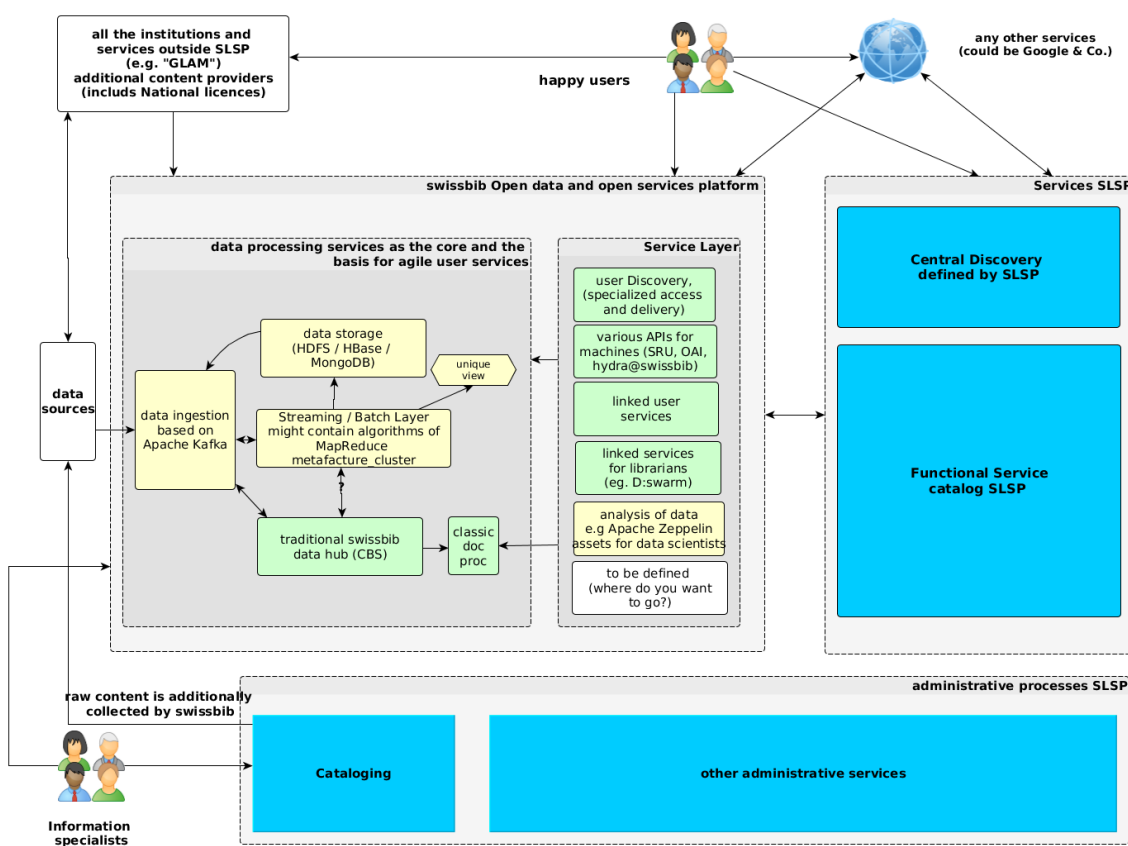


Abb. 18: schlanke data processing Lösung der swissbib Plattform im Zusammenspiel und in Integration mit SLSP⁶⁶
Quellenangabe: eigene Abbildung

Für diese Arbeit ist lediglich der Block „swissbib open data and open services“ relevant. Das Zusammenspiel mit der parallelen Initiative SLSP wird in diesem Kontext

⁶⁵ Für mehr Informationen s. <http://www.slideshare.net/pkiraly/metadata-quality-assurance-framework-at-qgml2016-full>

⁶⁶ Vgl. <https://blogs.ethz.ch/slsp/>

nicht besprochen. Da dieses Zusammenspiel jedoch eine Bedeutung für aktuelle Diskussionen im Bibliotheksumfeld hat, wird der Kontext zumindest visuell erweitert.

Der Block data processing services ist im Vergleich zu Abb. 5 sichtlich „entschlackt“ und erinnert wieder stärker an das einfachere ursprüngliche Modell aus dem Jahre 2013 wie in Abb. 1.

Dennoch sind alle Möglichkeiten und Ziele, wie sie in Kapitel 4 adressiert und beschrieben worden sind, enthalten.

- Der Event Hub für ein schnelles und einfacheres data ingestion sowie zur Integration von Diensten ist aufgenommen.
- Die Funktionen eines Batch-Layers auf Basis von metafactory_cluster mit der Möglichkeit eines alternativen data management zum Beispiel für clustering und de-duplication werden in den Streamprozessor integriert. Bisherige Hadoop MapReduce Funktionalität können im Batchmodus der Streamprozessoren ausgeführt werden. Ich gehe davon aus, dass ein Teil der Grundideen sowie der vorhandenen Implementierung von metafactory_cluster ohne grossen Aufwand übernommen werden kann.
- Vom ursprünglichen BatchLayer kann für die dauerhafte Persistierung HDFS als verteiltes Dateisystem zusammen mit HBase erhalten bleiben. Als Alternative zu dieser Variante ist eine NoSQL Datenbank wie MongoDB denkbar. Dies bis man erkennen sollte, dass sie den Anforderungen nicht mehr gerecht werden würde. Der Grund für eine solche Alternative: Wir haben bisher bereits über längere Zeit produktive Erfahrung mit diesem System, was die Einführung natürlich erleichtern würde.
- Es kann weiterhin ein Nebeneinander des bisherigen traditionellen Datenhubs zusammen mit der modernen und schnellen Streaming Variante geben. Die Gründe für diese Integration der bestehenden Komponente wurden bereits in Kapitel 4 genannt. Denkbar wäre auch, dass man Streaming Processor und CBS über einen zu entwickelnden Connector koppelt. Dies kann in beide Richtungen erfolgen, so dass CBS sowohl Datenquelle als auch Datensenke wäre.
- Der Streaming Prozessor kann sowohl Kafka als auch den persistenten storage als Datenquelle und/oder Ziel verwenden.

- Bei den Services für Menschen und Maschinen, die auf Basis dieses data processing Cores erbracht werden sollen/können, gibt es keinerlei Einschränkungen. Insbesondere können alle bereits in Kapitel 4 genannten Möglichkeiten und Ziele für eine zukunftsgerichtete und agile Informationsplattform adressiert werden.

Ziel von Arbeitspaket 4 war es, dass neben der allgemeinen Evaluation, auf dessen Basis ein erweiterter Architekturvorschlag wie in Abb. 18 entstehen sollte, auch eine erste Implementierung im Rahmen eines Prototyps analog Arbeitspaket 2 und 3 erstellt wird. Als Vorbereitung studierte ich über ein Wochenende die gerade erschienenen Bücher in [14] und [15] sowie das Online Apache Flink Training vor allem für den Bereich Batch processing⁶⁷. Als Testumgebung verwendete ich einen schnell zu benutzenden Stand Alone cluster auf meinem lokalen Laptop. Um eine bessere Vergleichsmöglichkeit mit den Arbeiten in Kapitel 5.2 (metafactory_cluster als BatchView) zu haben, setzte ich mir folgende Ziele:

- Anbindung von Kafka zur Abbildung eines Ladeprozesses in einen persistenten storage (im Rahmen des Projekts HBase)
- Verbindung mit HBase als data source. Dies als Grundlage für ein Clustering und zur weitergehenden Datenanalyse. Ein Schritt, der mir mit metafactory_cluster auf Basis der Implementierung des Jahres 2013 im Rahmen von Hadoop MapReduce Jobs nicht gelungen ist.

5.3.1 Prototyp: Anbindung von Kafka in Apache Flink

In diesem Prototyp ging es mir primär darum herauszufinden, wie leicht sich eine Kafka data source mit Flink einbinden lässt. Den weiteren Ladevorgang wie in Kap. 5.2 mit anschließender Transformation und Speicherung in HBase habe ich vor allem aus Zeitgründen nicht mehr angesehen. Da die entsprechenden Typen vorhanden sind, wäre dies im Rahmen eines Mappers, bei dem jede aus Kafka gelesene Informationseinheit zur Verfügung steht, kein Problem.

Ein wenig recherchieren musste ich bei der Auswahl der zur Verfügung stehenden Konnektoren, bis ich den passenden für eine Kompatibilität zwischen der jetzt aktuellen Flink Version mit der zuletzt erschienenen Kafka Version gefunden hatte. Beide

⁶⁷ Vgl. <http://dataartisans.github.io/flink-training/>

Projekte entwickeln sich sehr schnell, so dass nicht immer alle Bestandteile auf die letzten Versionen abgestimmt sind. Die erstellte Implementierung⁶⁸ erhebt nicht den geringsten Anspruch auf Schönheit. Sie dient lediglich als Grundlage für folgende Aussage:

Ein Streamprozessor wie Flink lässt sich einsetzen, um einen Ladeprozess aus Kafka mit anschließender Transformation und Speicherung in einem Storage zu ermöglichen. Damit ist der Grundstock für weitere Datenanalysen gelegt.

5.3.2 Prototyp: Anbindung von HBase in Apache Flink

Im erweiterten Lösungsvorschlag in Abb. 18 möchte ich MapReduce Analyse Algorithmen, wie sie mit metafactory_cluster für Hadoop MapReduce jobs entwickelt wurden in den Batchmodus von Apache Flink übertragen und weiterverwenden. Dafür muss jedoch eine Anbindung an HBase grundsätzlich möglich sein, da metafactory_cluster dieses Systems für die Persistierung verwendet⁶⁹. Der Prototyp⁷⁰ hat das ermöglicht. Ich konnte auf die im Rahmen des Arbeitspakts 3 in HBase gespeicherten Daten zugreifen und hatte sie zur weiteren Verwendung (für uns zur Zeit primär mit der Zielrichtung automatisiertes Clustering und Deduplizierung) im Rahmen eines Flink data source zur Verfügung.

6 Fazit / Ausblick

Bereits vor längerer Zeit (nicht erst im Rahmen dieser CAS Weiterbildung) habe ich mir vorgenommen, mich intensiver mit den Möglichkeiten des Frameworks metafactory_cluster auseinanderzusetzen. Ich habe darin die valable Möglichkeit gesehen, diese Lösung als Kern einer zukunftsorientierten data processing Umgebung für das Projekt swissbib einzusetzen. Das Problem was ich seinerzeit hatte: Es ist nicht einfach, mal eben ‚so aus dem Stand‘ die Ideen und Hintergründe alleine des Frame-

⁶⁸ Vgl. <https://github.com/guenterh/flinkTestRepository/blob/master/src/main/java/gh/FlinkKafka-Test.java>

⁶⁹ Falls man nicht, wie in der Diskussion von Abb. 18 bereits vorgeschlagen, ein anderes Storage-System wie z.B. MongoDB wählt. Dieses sollte als weit verbreitete Lösung in der Anbindung jedoch sehr viel einfacher als HBase zu handhaben sein.

⁷⁰ Vgl. <https://github.com/guenterh/flinkTestRepository/blob/master/src/main/java/gh/FlinkHBase-Test.java>

works zu verstehen. Kommt hinzu, dass die Software wesentliche Elemente des Hadoop Ecosystems einsetzt:

Hadoop MapReduce, HDFS, damit zusammenhängend alle Serverkomponenten eines Hadoop Clusters, HBase als Stagesystem.

Alleine für diese Auflistung lässt sich eine sehr lange Literaturliste erstellen. Man benötigt einige Zeit und auch eine gewisse Frustrationstoleranz, bis man anfängt, sich ein wenig sicherer nur in der Begriffswelt zu bewegen. Kommt hinzu, dass sich die Softwarekomponenten in diesem Ecosystem sehr schnell weiterentwickeln, was nicht dazu beiträgt, eine vor drei Jahren implementierte Software lauffähig zu bekommen, damit es einem leichter möglich ist, die Hintergründe zu erfassen. Ich denke dies sind die Hauptgründe dafür, warum es bis zu diesem CAS Kurs bei dem Vorsatz, sich mit dem Thema zu beschäftigen, geblieben ist.

Der Kurs hat dann geholfen, ein wenig Struktur in diese Unordnung zu bringen. Gleichzeitig erfährt man von ganz neuen Möglichkeiten, die eine Arbeit vor drei Jahren auf den ersten Blick schon wieder obsolet erscheinen lassen. Ich habe mich trotzdem auf dieses Thema eingelassen. Dies obwohl ich es im Juni für nicht unwahrscheinlich hielt, die Implementierung der für unser Projekt swissbib wesentlichen Teile, das automatisierte Clustern und die Deduplizierung von semistrukturierten Daten, lauffähig zu bekommen (was sich dann ja auch bewahrheitete). Ich wählte jedoch eine angepasste Strategie: Mit einem leicht angehobenen Wissensstand über neue Möglichkeiten verbreiterte ich den ‚Untersuchungsgegenstand‘. Ich hatte den Eindruck gewonnen, dass die Verfahren von 2013, in den Kontext der Diskussionsthemen des BigData Ecosystems von heute gestellt werden sollten. Damit wurden neben metafactory_cluster vor allem der Event Hub mit Kafka und Streaming Software zu weiteren Arbeitspaketen. Je mehr ich über Event Hubs im Allgemeinen und Kafka im Speziellen anfang zu lesen, desto mehr begann ich, über die aktuelle Architektur unseres swissbib Projekts nachzudenken. So sehr ich immer noch für die ursprünglich gewählte Architektur eintrete und sie als zukunftsfähig ansehe, ist bei genauerem Hinsehen doch erkennbar, dass sich mit den Ergebnissen der Entwicklung im Bereich BigData in den letzten Jahren einiges verbessern liesse. Aus diesem Grund habe ich ein umfangreicheres Kapitel zu Architekturüberlegungen im Vergleich jetzt / zukünftig hinzugefügt. Bleibt noch ein letzter Gedanke: Warum sollte man sich im Projekt swissbib eigentlich mit BigData im Speziellen und Daten im allgemeinen beschäftigen?

Geht es nicht darum, dass man den Kunden primär Services anbieten sollte (was das auch immer heissen mag) und anschliessend daran vielleicht noch ein wenig über die Daten, die man doch besitzt, nachdenken kann? Mittlerweile 6 Jahre produktiver swissbib Betrieb haben mich gelehrt, und die Informationen aus dem Kurs darin unterstützt, dass wir keinen einzigen unserer verschiedenen Services anbieten könnten, wenn wir nicht die richtigen Methoden und Verfahren (und damit sind vor allem die Möglichkeiten und die Kompetenz im Umgang mit der zur Verfügung stehenden Software gemeint) für ein modernes data processing (im Bibliotheksumfeld data management genannt) hätten. Diese Gedanken liessen den Titel der Arbeit entstehen.

Neben diesem eher allgemeinen Fazit am Ende einer Arbeit, welche Schlussfolgerungen ziehe ich ganz konkret aus den Ergebnissen für die zukünftige Informationsplattform swissbib und welcher Ausblick ergibt sich daraus?

- Die Komplexität der bestehenden workflows und damit auch der in den workflows eingesetzten Softwarekomponenten liesse sich durch die Einführung eines Event Hubs auf Basis Kafka sehr vereinfachen. Diese Massnahme lässt sich relativ kurzfristig umsetzen
- Die Einarbeitung in die Möglichkeiten von Streamprozessoren (sei dies nun Flink oder Spark oder durch das dann bereits vorhandene Kafka die Komponente ‚Kafka Streams‘) sollten fortgesetzt werden. Diese Form des data processing gibt den Institutionen die Möglichkeit und die Gestaltungsfreiheit, die Basis für Ihre so gesuchten Services zu schaffen. Das Zusammenspiel mit einem gut zu integrierenden domänenspezifischen Framework wie Metafactory, führt zu einem engen Zusammenspiel von eher SoftwareentwicklerInnen und Personen mit einem stärker informationswissenschaftlichen Hintergrund. Streaming Verfahren ermöglichen neue, im Bibliotheksumfeld bisher unbekannte, Services. Ein Beispiel ist die interaktive Datenanalyse durch Werkzeuge wie Apache Zeppelin.
- Es gibt immer noch viele Buzzwords im BigData Ecosystem und manches lässt sich nicht immer so einfach einsetzen. Dennoch haben wichtige Komponenten an Stabilität erheblich gewonnen und sind bereit für die breite Anwendung. Diese Chance sollte von den Institutionen (und vom Projekt swissbib) erkannt und genutzt werden.

7 Literaturverzeichnis

- [1] Shapira, Gwen. Introduction to Apache Kafka. O'Reilly, 2015. URL: <http://shop.oreilly.com/product/0636920038603.do>. Video
- [2] Kreps, Jay. I [Heart Symbol] Logs. First edition. Sebastopol, CA: O'Reilly Media, 2014. Print.
- [3] Marz, Nathan, and James Warren. Big Data : Principles and Best Practices of Scalable Real-Time Data Systems. Shelter Island, NY: Manning, 2015. Print.
- [4] Shapira, Gwen, and Palino Todd. Kafka The Definite Guide: Real-Time Data and Stream Processing at scale. First edition [Early Release]. Sebastopol, CA: O'Reilly Media, 2016. Print
- [5] Dean, Alexander. Unified Log Processing: Integrating and processing Event Streams Version 12 MEAP. Shelter Island, NY: Manning, 2016. Print
- [6] Markus Michael Geipel, Christoph Böhme, and Jan Hannemann. "Metamorph: A Transformation Language for Semi-Structured Data." N.p., May/June 2015. Web. 2 Sept. 2016. URL: www.dlib.org/dlib/may15/boehme/05boehme.html
- [7] Christoph Böhme, and Pascal Christoph. "Analysis and Transformation of Library Metadata with Metafacture - Workshop, Swib 2015." N.p., n.d. Web. 2 Sept. 2016. URL: <http://swib.org/swib14/programme.php> / http://swib.org/swib14/slides/christoph_swib14_40.zip
- [8] Christoph Böhme. "Das Metamorph-Datenmodell." N.p., n.d. Web. 2 Sept. 2016. URL: <http://b3e.net/metamorph-book/latest/datamodel.html>
- [9] "Wiki Metafacture-Cluster." N.p., n.d. Web. 3 Sept. 2016. URL: <https://github.com/culturegraph/metafacture-cluster/wiki>
- [10] George, Lars, HBase : The Definitive Guide. Sebastopol, CA: O'Reilly Media, 2016. Print.
- [11] O'Dell, Jean-Marc Spaggiari, Kevin. Architecting HBase Applications. N.p. shop.oreilly.com. Web. 3 Sept. 2016.
- [12] Dimiduk, Nicholas, Khurana, Amandeep. HBase in Action. Shelter Island, NY: Manning .November 2012. Print
- [13] Miner, Donald, and Adam Shook. *MapReduce Design Patterns*. Sebastopol, CA: O'Reilly, 2013. Print.

- [14] Wadkar, Sameer, and Rajaram, Hari. Flink in Action, Version 1 MEAP. Shelter Island, NY: Manning, 2016. Print
- [15] Friedmann, Ellen, and Tzoumas Kostas. Introduction to Apache Flink: Stream Processing for Real Time and Beyond. Sebastopol, CA: O'Reilly, 2016. Print.

Erklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe. Alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten und nicht veröffentlichten Schriften entnommen wurden, sind als solche kenntlich gemacht. Die Arbeit ist in gleicher oder ähnlicher Form oder auszugsweise im Rahmen einer anderen Prüfung noch nicht vorgelegt worden.

Basel, den 9.10.2016

Günter Hipler