A Lightweight Workflow Management Framework for Scientific Research Data Management

Canadian Research Software Conference 2022

Ian Percel and Xiaofang (Ada) Xing

University of Calgary, Research Computing Services

May 31, 2022

Development Goal - Problem

The goal of this research is to develop a Scientific Research Data Management workflow framework. An SRDM Workflow:

1. connects instrument computers or source databases to scalable processing infrastructure (e.g. HPC Cluster)
2. interleaves light data management processing tasks with processing tasks that can take advantage of parallel resources
3. has large volumes and complex (domain specific) formats
4. includes integration of data into a user-facing database
5. complex parts of the pipeline reside in a single security domain

typified by VM data management of a network of instruments and submission of computationally intensive tasks to a job scheduled (e.g. SLURM) HPC cluster

Development Goal - Solution Requirements

- infrastructure modular - loose coupling of customizable components (e.g. user facing database) to the overall framework
- workflow modular (composition of reusable, optimized data pipeline patterns)
- portable and infrastructure-light
- CLI oriented
- supports single-security domain distributed task scheduling

Loose-coupling here is in contrast with a complex and rigid dependence on the presence of specific schema and APIs.

Influencing Work

1. FAIRly Big Pipeline Framework (Wagner et al. 2022)
2. C-Lab SEA tool for Hardware-Software Co-Design (Tacken et al. 1999)
3. State Machine Workflow Nets (Corro Ramos et al. 2006)
4. Turbine (Wozniak et al. 2012)
5. Pegasus (Deelman et al. 2001-2019)

Software Model for Framework Solution

The overall framework can be implemented in four components (only last two needed at runtime)

1. Task Pre-Compiler = transforms high-level model into sub-workflows for independent scheduling
2. Task Compiler
3. Workflow Executor
4. Trace History Communication Protocol

Software Model for Framework Solution - Task Precompiler

Inputs:

1. full abstract workflow graph
2. sub-workflow code and specifications for time, compute resource requirements, memory requirements
3. sub-workflow specification of input traces

Output:

1. collection of merged code, new job specifications and submit logic produced by aggregating subworkflows and estimating timing
2. expected output traces

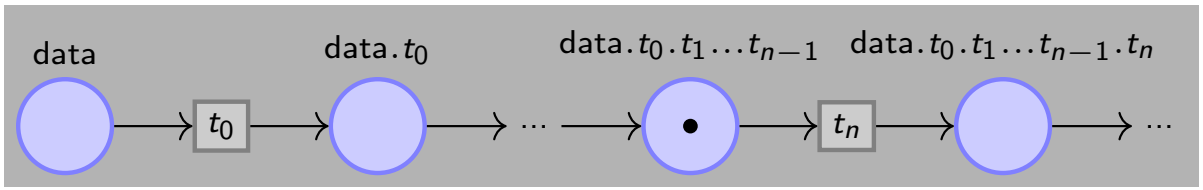
Which tasks can be scheduled as combined jobs to an HPC cluster? How do we make them run asynchronously so that we can benefit from pipeline parallelism?

High-level Petri Net Model

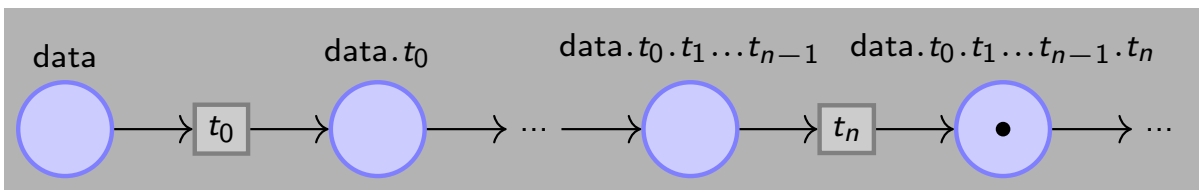
Directed bipartite graph of places and transitions

- Places = preconditions for task execution
- Transitions = tasks with complex operations and timings
- Labelled arcs = communication channels
- Labelled markings = distributed state, changes under transition firing/task execution

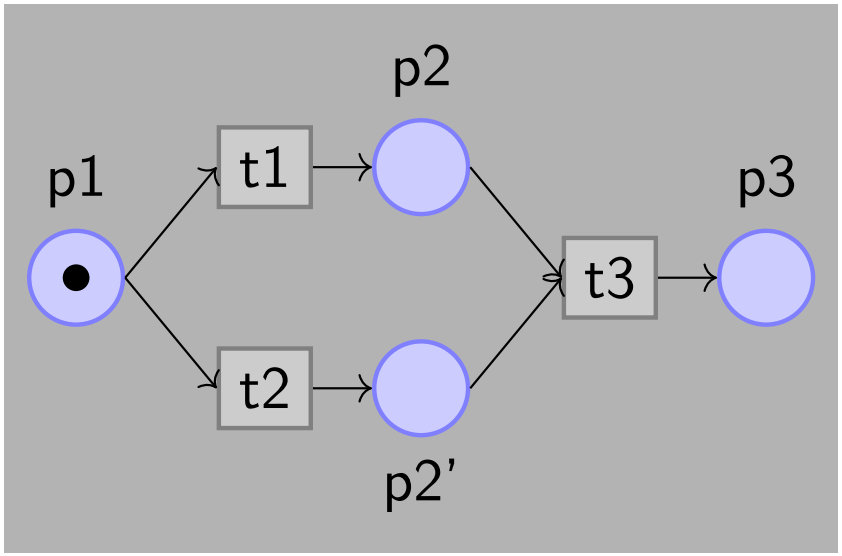
Places = Partial Histories of Traces



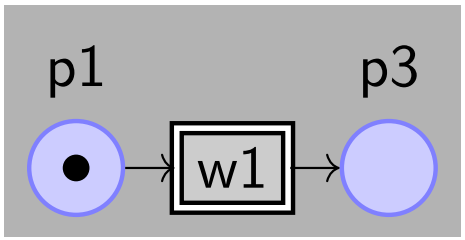
t_n fires means that, in a graph edge path notation for function application, $data.t_0.t_1 \ldots t_{n-1}$ becomes $data.t_0.t_1 \ldots t_{n-1}.t_n$
as the $n+1$th processing step on the data



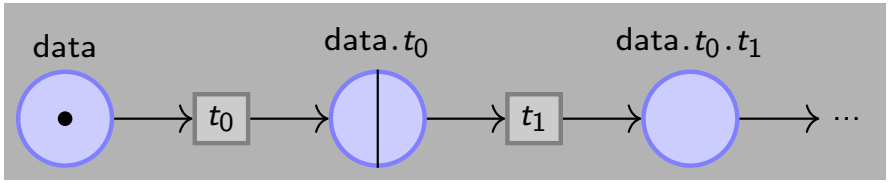
High-level Petri Net Model: Subnet Reduction



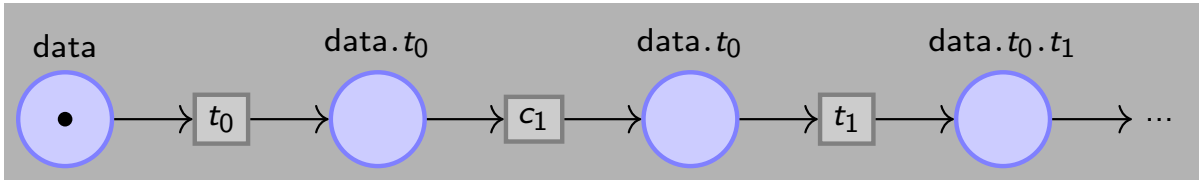
reduction by replacing a subnet with an agglomerated transition



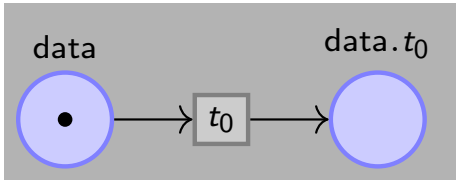
High-level Petri Net Model: Place Cuts



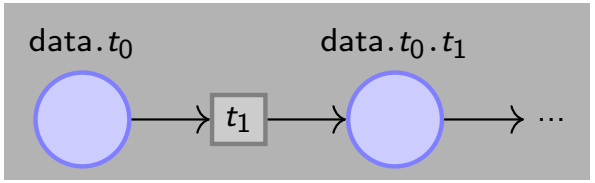
p_1 split yields two places with identical conditions connected by a communication transition



Alternately, as a partitioning, we see these as two independently scheduleable workflows with implicit communication about local state

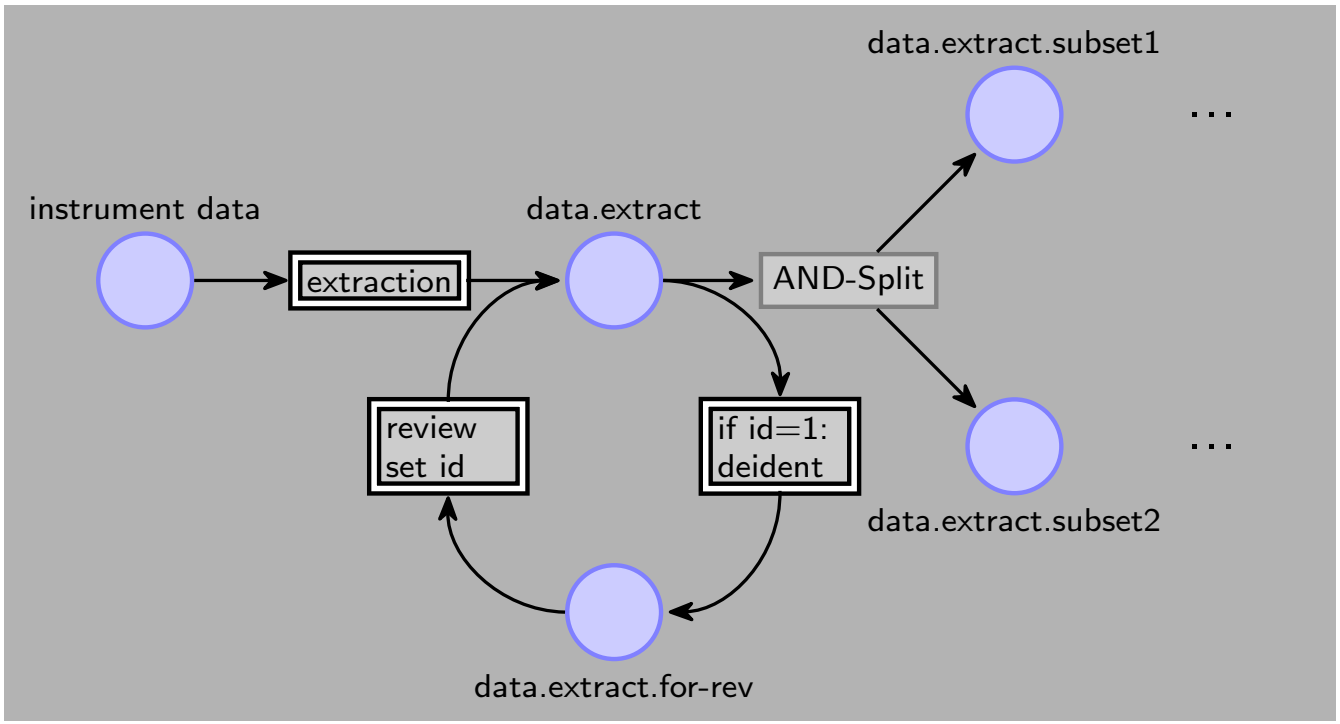


process 1 on server 1

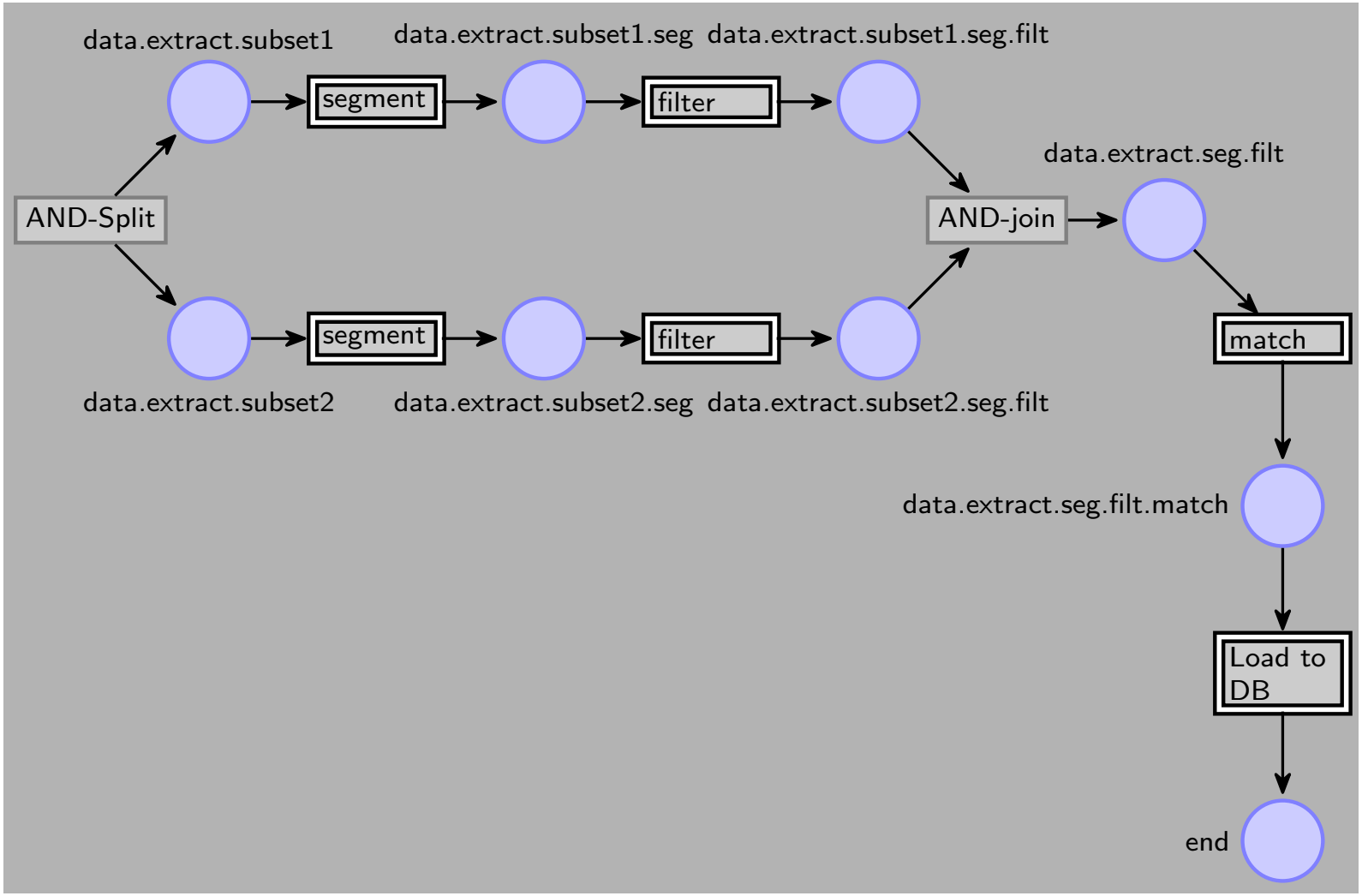


process 2 on server 2

A Typical Wf Net part 1



A Typical Wf Net part 2

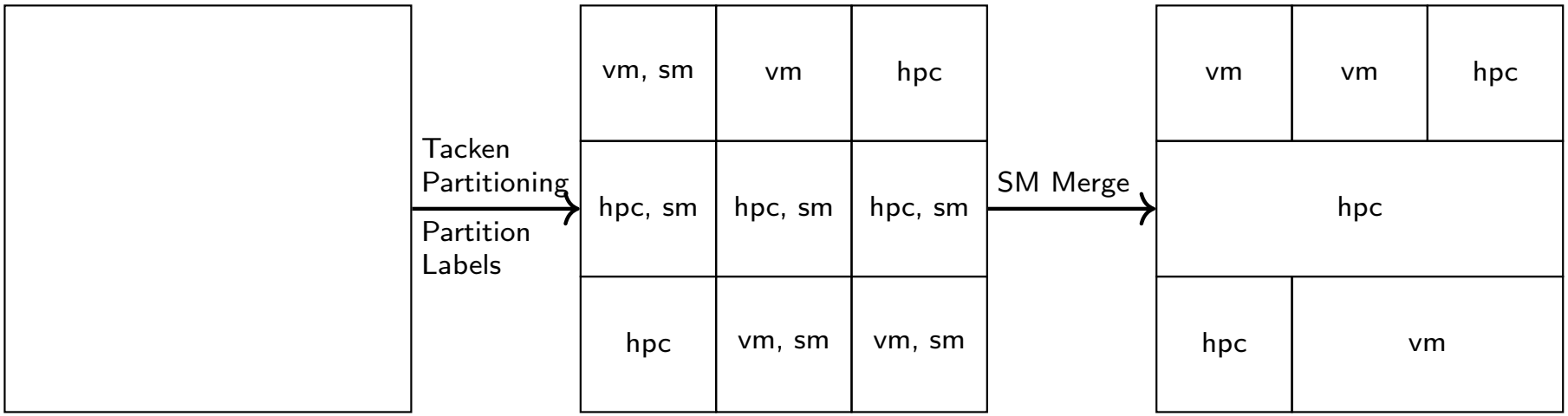


Partitioning Algorithm for Pre-Compiler

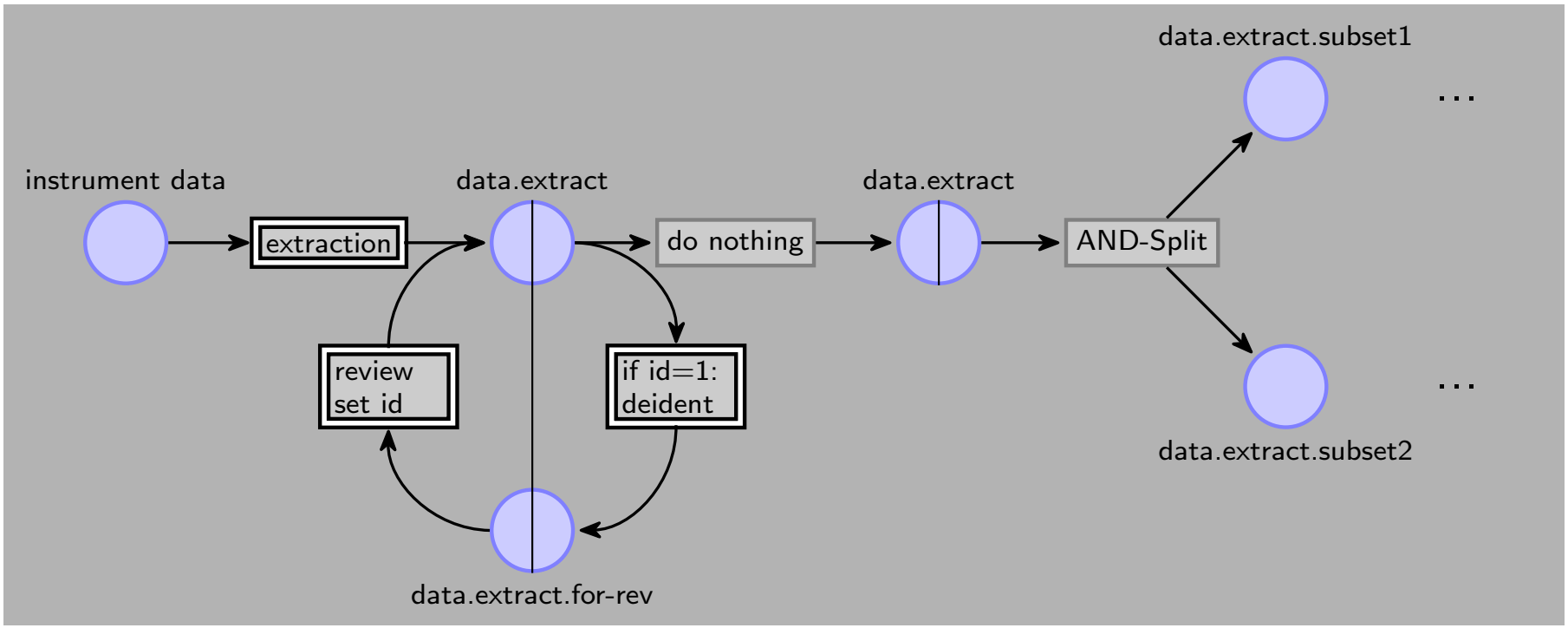
Requirements for a useful Partitioning Algorithm for the Pre-Compiler

1. Minimize communication
2. Asynchronously schedule workflow partitions to different resources
3. Maximize benefit from using parallel resources
4. Compute partitioning quickly

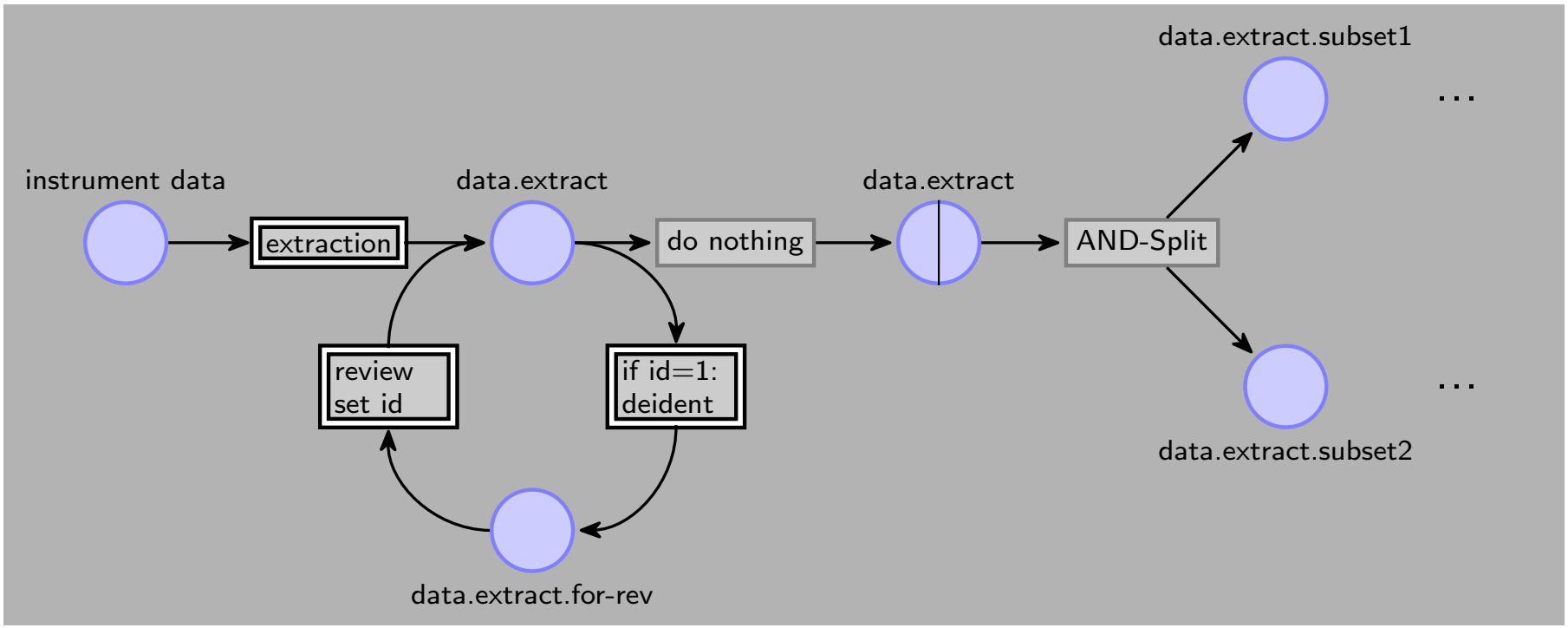
Hierarchical SMWf Net Partitioning Algorithm



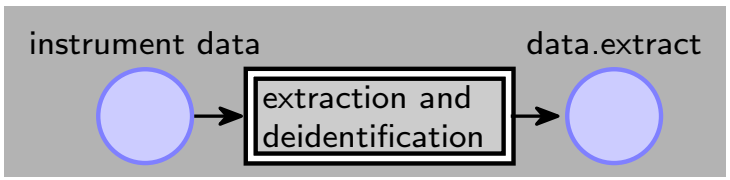
A Typical Wf Net under Hierarchical SMWf Partitioning Algorithm: FCC/BCC Split

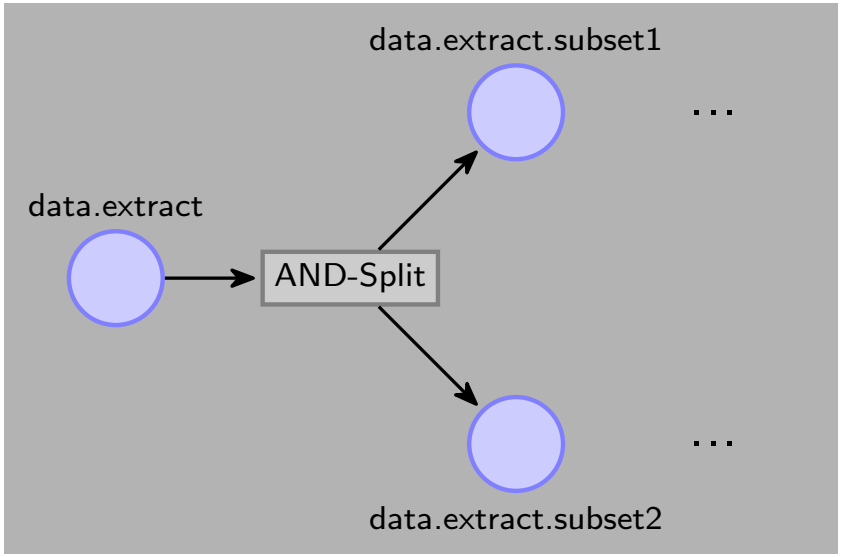


A Typical Wf Net under Hierarchical SMWf Partitioning Algorithm: non-reentrant merge of SMs

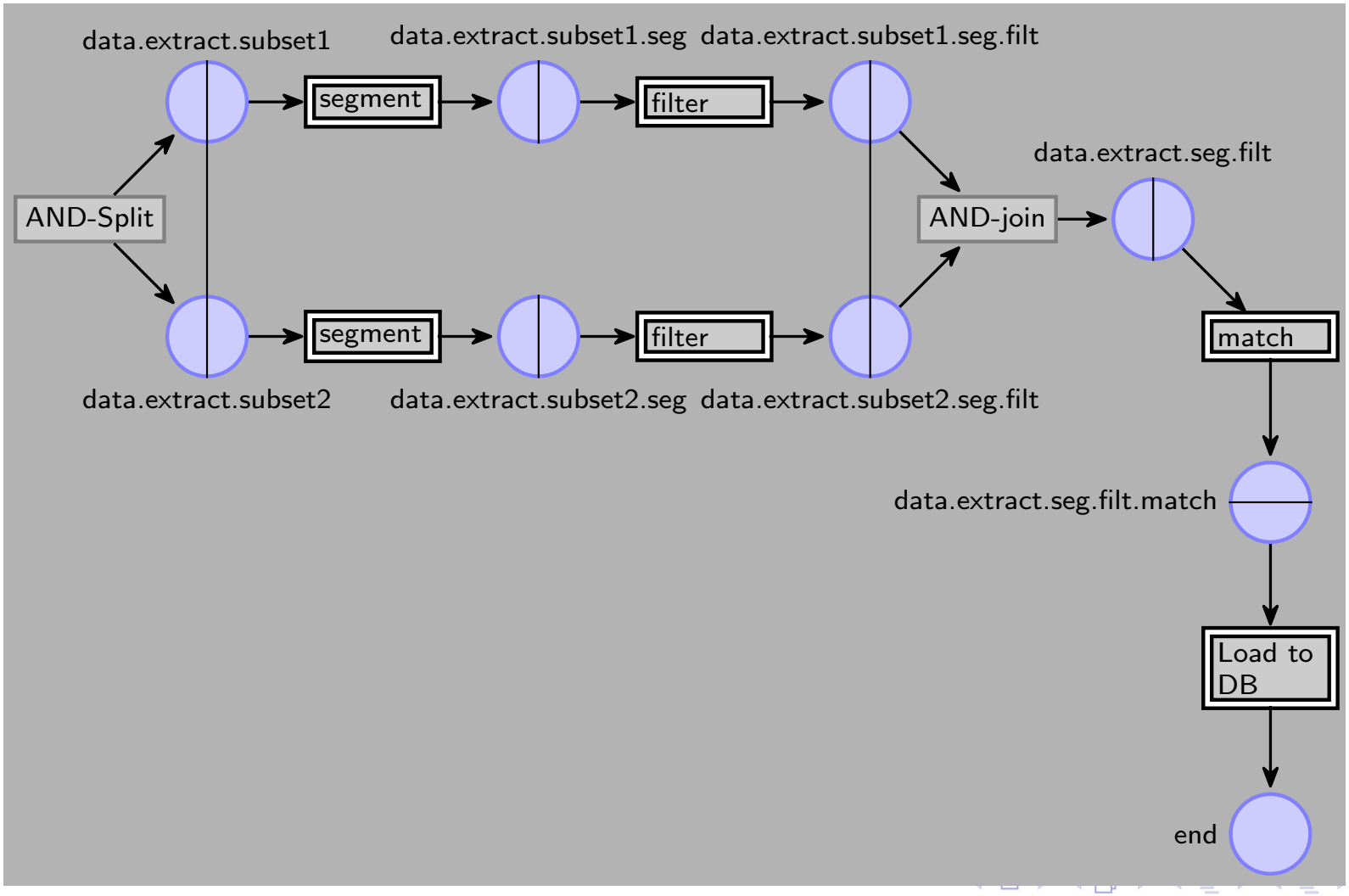


A Typical Wf Net under Hierarchical SMWf Partitioning Algorithm: sub-Wf substitution

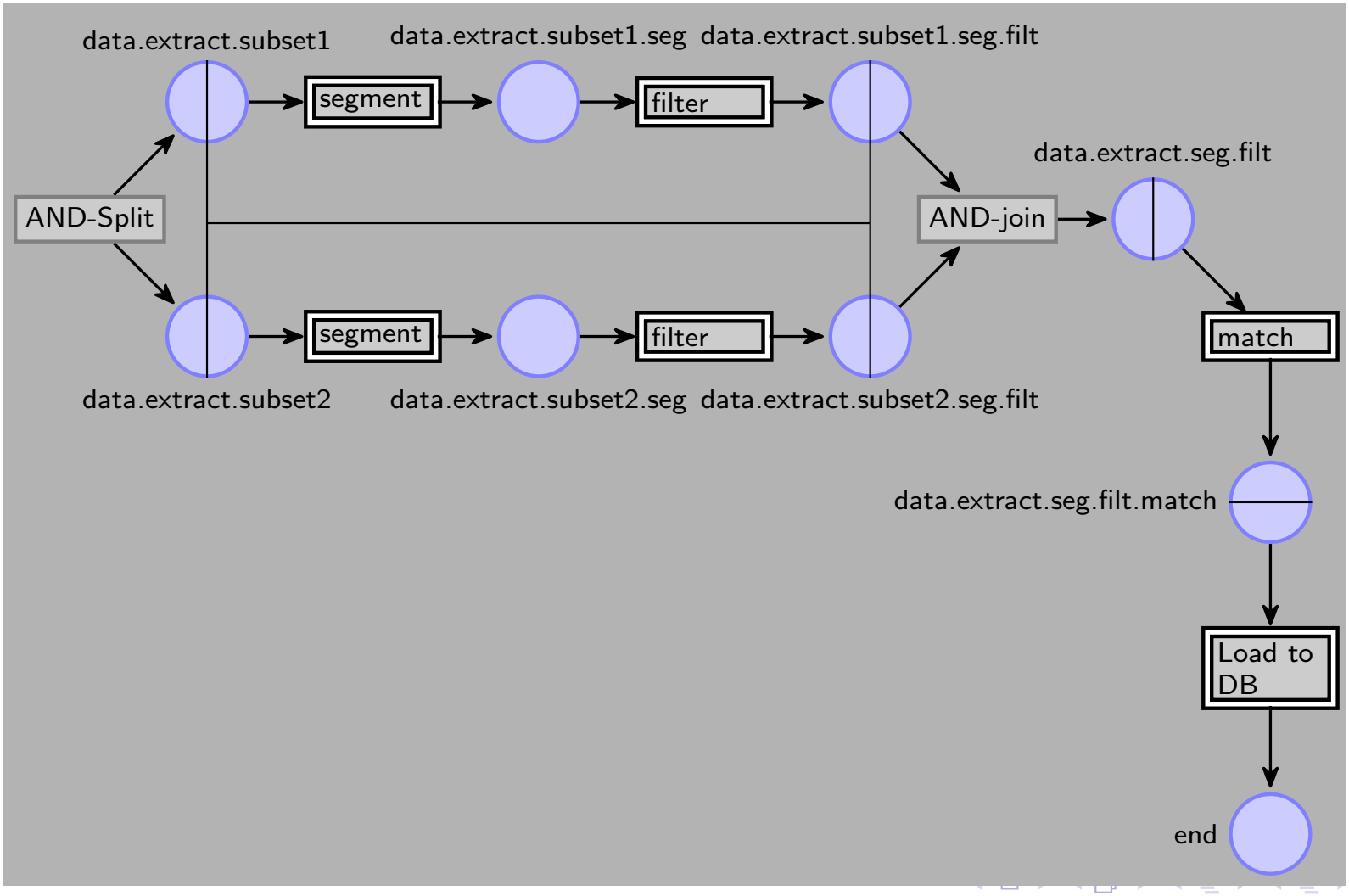




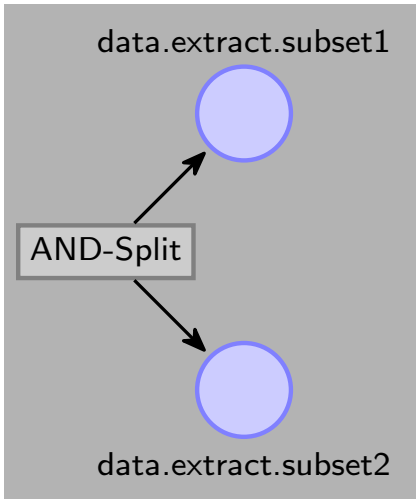
A Typical Wf Net under Hierarchical SMWf Partitioning Algorithm: FCC/BCC Split

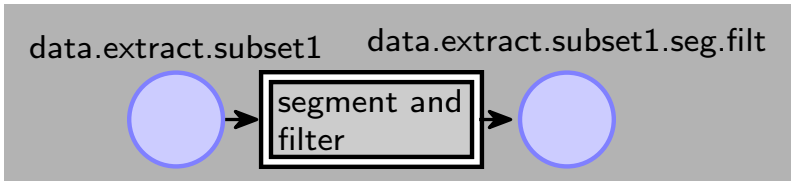


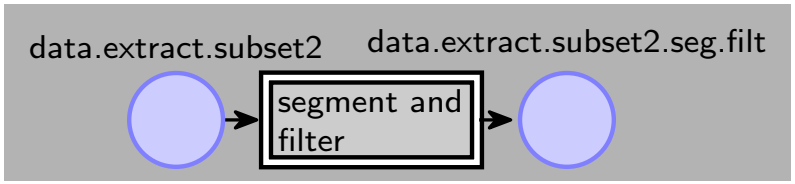
A Typical Wf Net under Hierarchical SMWf Partitioning Algorithm: Possible SM Merge

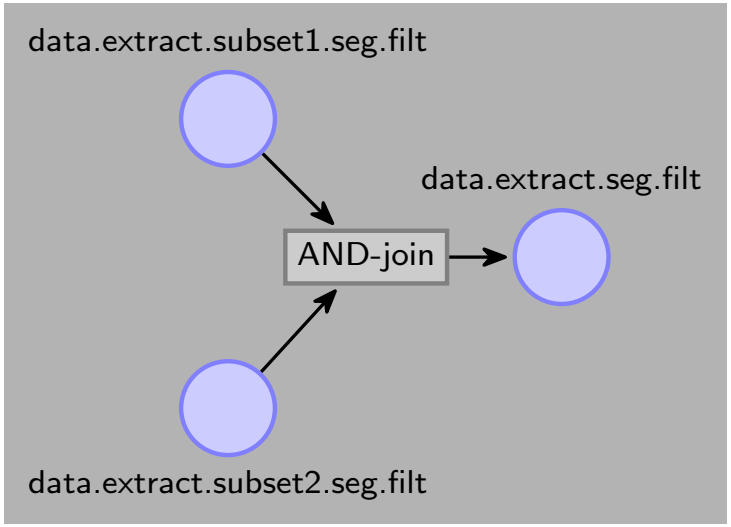


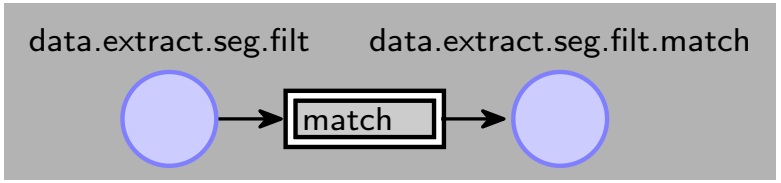
A Typical Wf Net under Hierarchical SMWf Partitioning Algorithm: Possible SM Merge and sub-Wf Substitution

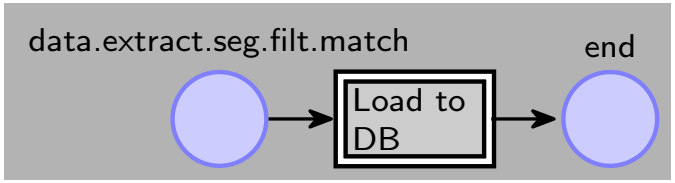












git-annex, DataLad, and FAIRly Big

- Lean clones for process provenance (trace history) communication
- Ephemeral workspaces enable reproducibility but also are natural compilation targets
- Concurrency limitations on the provenance record push stem from git-annex design
- Since work is always in non-overlapping branches, no merge conflict should be possible

Impact of Decoupling User Facing Databases from Workflow Process Management

Some User Facing Metadata Management Strategies

1. Custom Relational Database
2. Laboratory Information Management Development System (LabKey)
3. Clinical Study Survey Database (REDCap)

Can one tool solve all of the problems that these different tools were individually designed to solve? No.

Variation in Data Use and Data Type Observed in Research Projects

Data Management Supports Needed

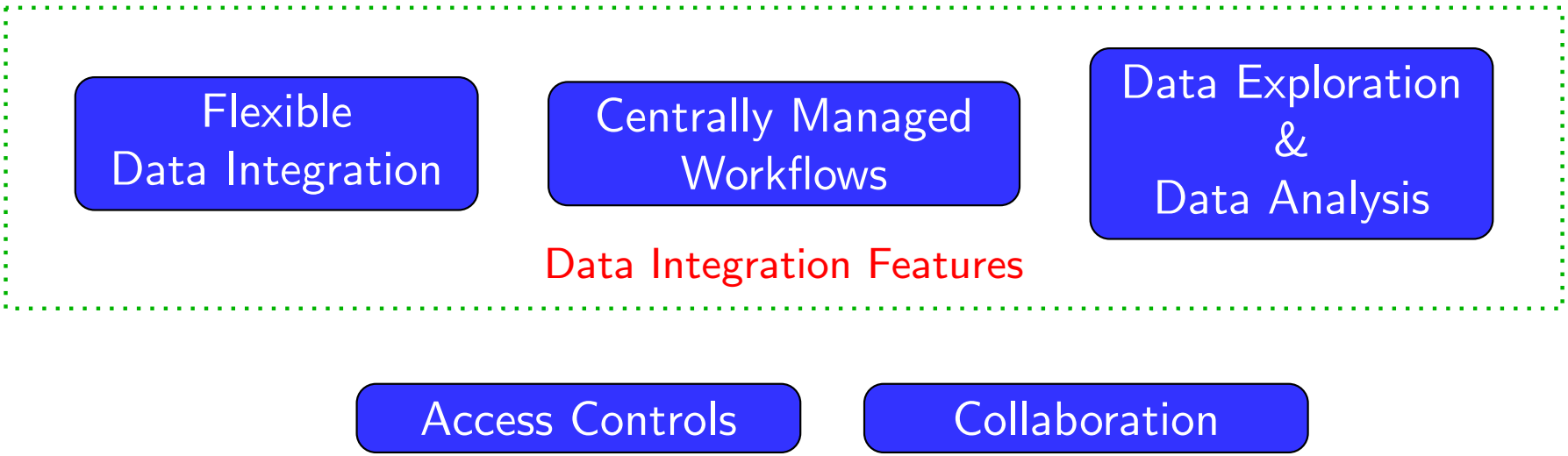
- Processing Workflow
- Quality Control Workflow
- Ingestion into Database

Data Type

- Multi-Omics data
- Imaging data
- Study Metadata

We will examine using LabKey for the case of handling Multi-Omics Data and Study Metadata in a Quality Control Workflow.

LabKey Workflow-Oriented Architecture



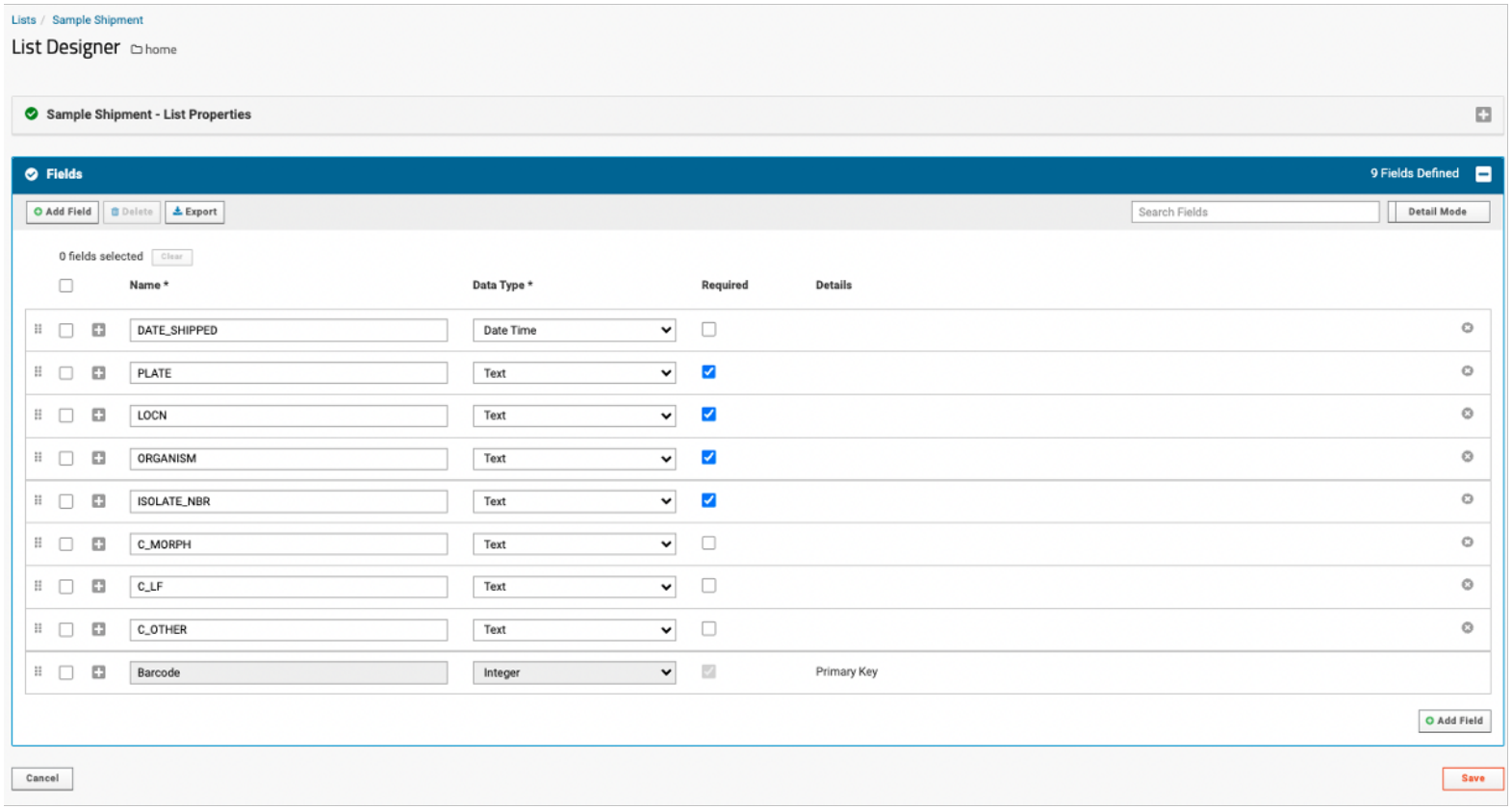
LIMS Specific Example: Sample Tracking

A research group faces a problem comparing shipping registers (below) to plate locations used in biological assays

	A	B	C	D	E	F	G	H	I
1	DATE_SHIPPED	PLATE	LOCN	ORGANISM	ISOLATE_NBR	C_MORPH	C_BH	C_OTHER	Barcode
17	2021-08-23	SA064	B,6	MRSA	BI_07_2273	NORM	2		338340112
18	2021-08-23	SA064	B,7	MRSA	BI_07_2274	NORM	1		338346126
19	2021-08-23	SA064	B,8	MRSA	BI_07_2275	NORM	2		338346111
20	2021-08-23	SA064	B,9	MRSA	BI_07_2276	NORM	1		338341258
21	2021-08-23	SA064	B,10	MRSA	BI_07_2277	NORM	2		338346270
22	2021-08-23	SA064	C,1	MRSA	BI_07_2278	NORM	2		338342403
23	2021-08-23	SA064	C,2	MRSA	BI_07_2279	NORM	2		338346135
24	2021-08-23	SA064	C,3	MRSA	BI_07_2280	NORM	2		338340124
25	2021-08-23	SA064	C,4	MRSA	BI_07_2281	NORM	2		338354046
26	2021-08-23	SA064	C,5	MRSA	BI_07_2282	NORM	2		338353993
27	2021-08-23	SA064	C,6	MRSA	BI_07_2283	NORM	1		338339973
28	2021-08-23	SA064	C,7	MRSA	BI_07_2284	NORM	2		338340143
29	2021-08-23	SA064	C,8	MRSA	BI_07_2285	NORM	2		338342430
30	2021-08-23	SA064	C,9	MRSA	BI_07_2286	NORM	2		338342466
31	2021-08-23	SA064	C,10	MRSA	BI_07_2287	NORM	2		338346162
32	2021-08-23	SA064	D,1	MRSA	BI_07_2288	NORM	1		338346147
33	2021-08-23	SA064	D,2	MRSA	BI_07_2289	NORM	2		338342452
34	2021-08-23	SA064	D,3	MRSA	BI_07_2290	NORM	2		338347227
35	2021-08-23	SA064	D,4	MRSA	BI_07_2291	NORM	2		338340088
36	2021-08-23	SA064	D,5	MRSA	BI_07_2292	SCV	1		338354030
37	2021-08-23	SA064	D,6	MRSA	ATCC_43300	NORM	2		338354052
38	2021-08-23	SA064	D,7	MRSA	BI_07_2293	NORM	2		338347243
39	2021-08-23	SA064	D,8	MRSA	BI_07_2294	NORM	2		338354050
40	2021-08-23	SA064	D,9	MRSA	BI_07_2295	NORM	2		338341255
41	2021-08-23	SA064	D,10	MRSA	BI_07_2297	NORM	2		338341299
42	2021-08-23	SA064	E,1	MRSA	BI_07_2298	NORM	1		338342444
43	2021-08-23	SA064	E,2	MRSA	BI_07_2299	NORM	2		338342833
44	2021-08-23	SA064	E,3	MRSA	BI_07_2300	NORM	2		338340096
45	2021-08-23	SA064	E,4	MRSA	BI_07_2301	NORM	1		338341253
46	2021-08-23	SA064	E,5	MRSA	BI_07_2302	NORM	1		338346128
47	2021-08-23	SA064	E,6	MRSA	BI_07_2303	NORM	2		338341251
48	2021-08-23	SA064	E,7	MRSA	BI_07_2304	NORM	1		338340157
49	2021-08-23	SA064	E,8	MRSA	BI_07_2305	NORM	1		338340094
50	2021-08-23	SA064	E,9	MRSA	BI_07_2306	NORM	1		338351690
51	2021-08-23	SA064	E,10	MRSA	BI_07_2307	NORM	2		338340148
52	2021-08-23	SA064	F,1	MRSA	BI_07_2308	NORM	2		338340080
53	2021-08-23	SA064	F,2	MRSA	BI_07_2309	NORM	1		338342465
54	2021-08-23	SA064	F,3	MRSA	BI_07_2310	SCV	1		338341241
55	2021-08-23	SA064	F,4	MRSA	BI_07_2311	NORM	2		338341305

LIMS Specific Example: Sample Tracking

In Labkey, this can be solved readily using a new list design and connecting it with Assay data by custom queries



Questions

Motivating Observations from Petri Net Literature

By Carro Ramos et al 2006 "Synthesis and reduction of state machine workflow nets", state machine workflow nets provide a nice class of nets to use for individual partitions because they

- include linear sequences and exclusive choice
- do not include parallel splits

state machine workflow nets (i.e. those consisting of only transitions with one incoming and one outgoing edge) can be easily synthesized from linear extension (refinement) and addition of an exclusive choice (addition).

Motivating Observations from Petri Net Literature 2

Tacken et al 1999 "A Method for Prepartitioning of Petri Net Models for Parallel Embedded Real-Time Systems", provides a good initial partitioning that can be obtained by forming minimal Backward Conflict Complete (BCC) and Forward Conflict Complete (FCC) subsets by refining transitions into two so that every transition has only one incoming or one outgoing edge and the place separating them can be the site of a place cut. These partitions have some useful properties

- Places that are cut only connect two partitions, never more
- Communication across place cuts can always take a send and forget form
- State machines are both BCC and FCC at every transition and so always cut into state machines
- The merging of two partitions produced by the method of Tacken et al forms a state machine if and only if both partitions are state machines

since place cuts always connect only two partitions, communication can't fail to trigger due to async partition scheduling. The only risk is with re-entrant connections.

Hierarchical SMWf Net Partitioning Algorithm

1. Produce a pre-partitioning following the algorithm of Tacken et al.
2. Label each partition based on the resource requirements of the included transitions
3. Identify all partitions that are state machines
4. Beginning from the non-state machine partitions, attempt to merge adjacent partitions (without crossing a resource label boundary) until a workflow net is formed so that there is only one initial and one terminal transition. If this can be done, replace the merged partition with a single sub-workflow. This is now a state machine for the purposes of the algorithm.
5. Examine all non-reentrant merges of state machine partitions and compute timing estimates for asynchronously scheduling the partitions under the assumption that (a) all residual (non-state machine) partitions are merged up to label boundaries or (b) no residual partitions are merged
6. Select the time minimizing partitioning

Hierarchical SMWf Net Partitioning Algorithm Properties

1. Due to the communication properties of place cuts in Tacken et al, deadlocking on communication isn't possible so long as job start is on any non-empty marking
2. State Machine partitions are cheap to identify (it is an explicit structural property of transitions) and all contiguous merges of SM partitions are again state machines
3. A reentrant relationship between two (or more) partitions comes from splitting loops or splitting split-join pairs across partitions and so excluding these cases is not computationally expensive
4. Pipelining alternatives for each parallel branch are considered but merging branches into a single job is not considered
5. Since the jobs run on local state information only, pipeline parallelism, data parallelism, and checkpoint-restart at the partition scale are intrinsically enabled
6. For all SMWf nets, liveness and boundedness are left intact