

Building a

Lightweight Distributed Computing System for LDMX

Lene Kristian Bryngemark, Stanford University

for LKB, David Cameron, Valentina Dutta, Thomas Eichlersmith, Balazs Konya, Omar Moreno, Geoffrey Mullier, Florido Paganelli, Ruth Pöttgen, Fuzzy Rogers, Andrii Salnikov, and Paul Weakliem (submitted paper: [arXiv:2105.02977](https://arxiv.org/abs/2105.02977))

vCHEP 2021, May 17-21 2021



Large-scale computing for a small-scale collaboration

Researcher's home cluster

- easy to get access
- fast to deploy software
- tends to be manual labor intensive
 - job monitoring
 - book keeping is difficult
- collaborator access requires local user accounts across institutes
 - or a central service
 - and/or data transfer

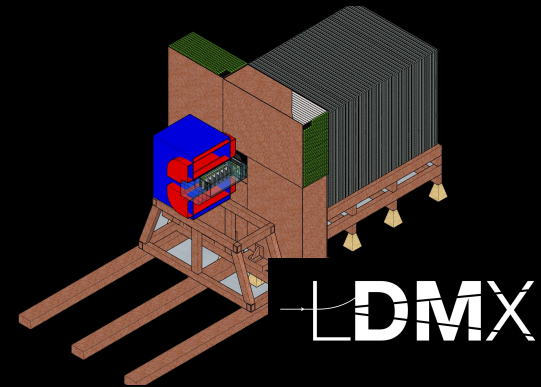
Distributed computing system

- pools collaboration resources
- scalable
- avoids knowledge/technology lock-in
- takes a lot of work and resources to set up and operate
 - ... or does it?

Example collaboration: Light Dark Matter eXperiment

LDMX: ~30 collaborators across 9 institutes

- planned fixed-target electron beam experiment
- missing-momentum search for dark matter
- extensive simulations needed for detector design
 - simulation framework: a Geant4 application
 - Python interface for configuration
 - ROOT for persistency



SLAC is the host lab

- central computing resources and long-term data storage
 - easy for researchers to obtain SLAC computing accounts for access
 - where simulations were initially run

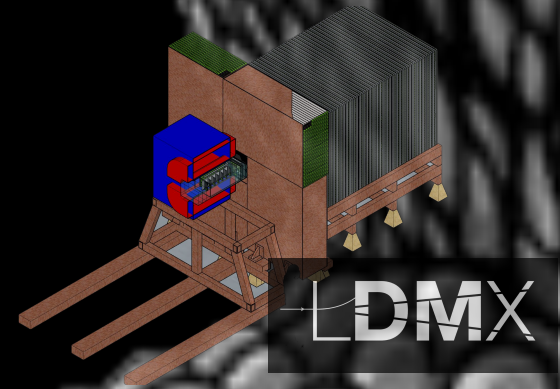
Example collaboration: Light Dark Matter eXperiment

LDMX: ~30 collaborators across 9 institutes

- planned fixed-target electron beam experiment
- missing-momentum search for dark matter
- extensive simulations needed for detector design
 - simulation framework: a Geant4 application
 - Python interface for configuration
 - ROOT for persistency

SLAC is the host lab

- central computing resources and long-term data storage
 - easy for researchers to obtain SLAC computing accounts for access
 - where simulations were initially run



LDMX distributed computing strategy

Utilize resources at other collaborating institutes

- mainly CPU hours but also storage
- solve access or data transfer between institutes and to SLAC
- not enough resources to build a system from scratch

Strategy: leverage established LHC distributed computing technologies

- ARC * for job submission and data transfer
- Rucio for cataloging data (and metadata)

4 participating LDMX institutes:

- Caltech, UCSB, SLAC (US)
- Lund (Sweden)

Participating sites provide:

- a batch system (so far SLURM / HTCondor)
- local storage (~10 TB) on shared file systems

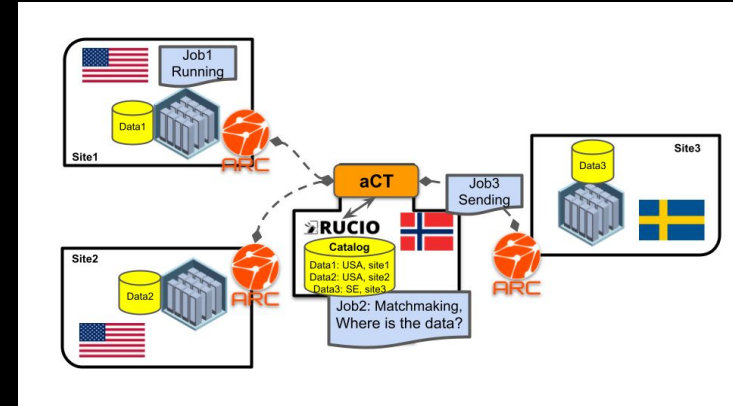
Central services in Oslo (Norway)

SLAC provides larger central storage

* Advanced Resource Connector

the Lightweight Distributed Computing System (LDCS)

- an ARC Computing Element runs at each site
 - submits batch jobs for authenticated LDMX user
 - transfers data to/from storage
- security layer built around X.509
- submission done from one ARC Control Tower
 - application-facing side where user defines:
 - simulation specifics such as software tag, input data set needed, and job steering file
 - production specifics such as walltime and number of jobs
 - resource-facing side:
 - monitors job status at sites
 - submits jobs according to queue depth
 - resubmits failed jobs
 - data-driven job brokering if input file needed



Key advantages over pure batch:
No need for direct user access to cluster

Takes data handling out of batch job:

- not part of user code
- CPUs not idle while transfers happen

LDMX specific solutions for LDCS

LDMX software

- built along all dependencies into a Docker container
 - a new ldmx-sw release triggers automatic GitHub action to build new container
- set up with Singularity on the sites

Rucio catalog

see later talk by M. Barisits

- output data location (used for job brokering)
- book keeping of simulation settings
 - using Rucio's generic JSON format option
 - parses ldmx-sw printout of all configuration settings

Automation of book keeping

Enforces reproducibility:

- sw version frozen
- all simulation settings cataloged

Prometheus + Grafana monitoring

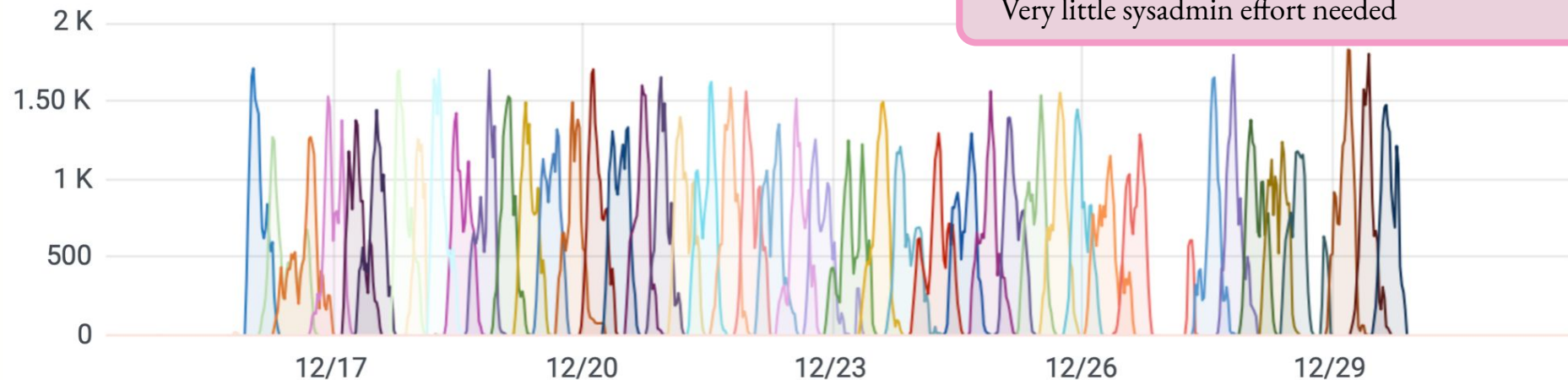
Features and lessons learned: example campaign Dec 2020

Submitted 225 000 jobs*, in 45 batches of 5000 each

- on average 1000 jobs running simultaneously (saturates capacity)
- output file size ~250 MB, all uploaded to SLAC using GridFTP
- site sysadmins reported ~1h of work, including setup and interventions

Running jobs ▾

Higher throughput than at SLAC batch only
Very little sysadmin effort needed



* constituting one simulation sample, i.e. identical settings except for random number

Features and lessons learned: example campaign Dec 2020

aCT resubmits failed jobs based on batch job exit codes *

- rate of unsuccessful jobs dropped from 44% in the first test campaign (May 2020) to 0.03%



* good choices include: node network glitches, hitting walltime on slower nodes, queue preemption

Features and lessons learned: example campaign Dec 2020

Last step is output transfer to SLAC and Rucio registration

- two occasions where transfer was stalled by local problem at SLAC
- ARC able to hold output until resolved



Still more to be learned

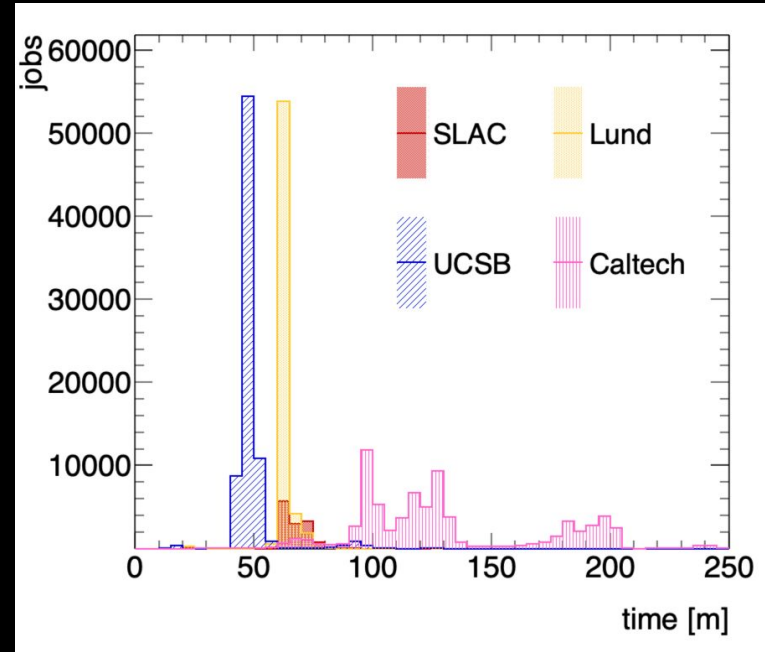
A rich source of job and batch running metadata

- can learn from any job metadata we keep

Next steps involve

- setting up a user friendlier submission interface
- user access to produced simulation data for analysis jobs through LDCS (instead of only via SLAC)

example: walltimes extracted from Rucio

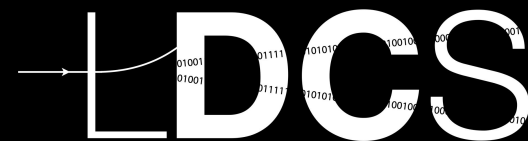


Summary

Even for a small-scale experiment, significant gains in going to distributed computing.

LDCS has shown that we can

- leverage existing technologies to achieve a lightweight setup
- improve scalability
- improve resource sharing and data access
- improve reproducibility and simulation book keeping.



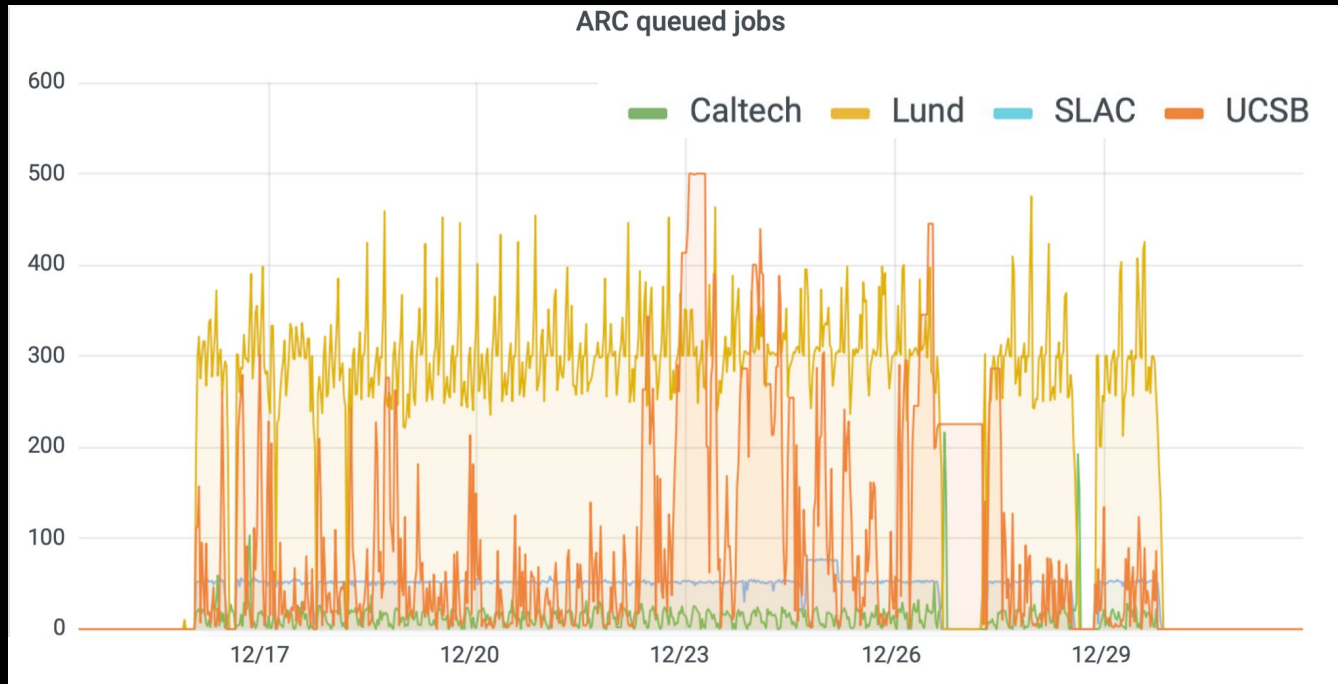
Much more detail in the [submitted paper](#)

Backup

More from the example campaign Dec 2020

ARC keeps its own queue of jobs assigned to each site

- no time wasted between one job finishing and pushing the next to batch queue



More from the example campaign Dec 2020

Script checks system occupancy using ARC monitoring, automatically submits more

- implemented here, before then, production manager submitted new batch



Security layer with X.509 proxy credentials

Authentication of production user through robot certificates

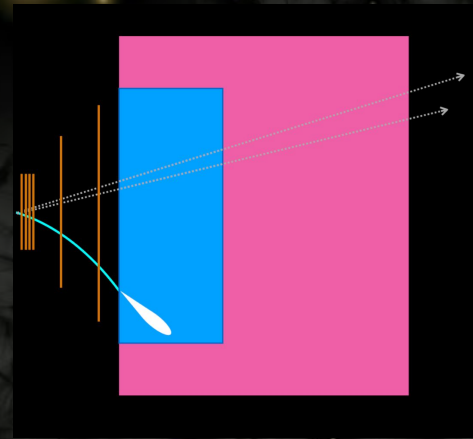
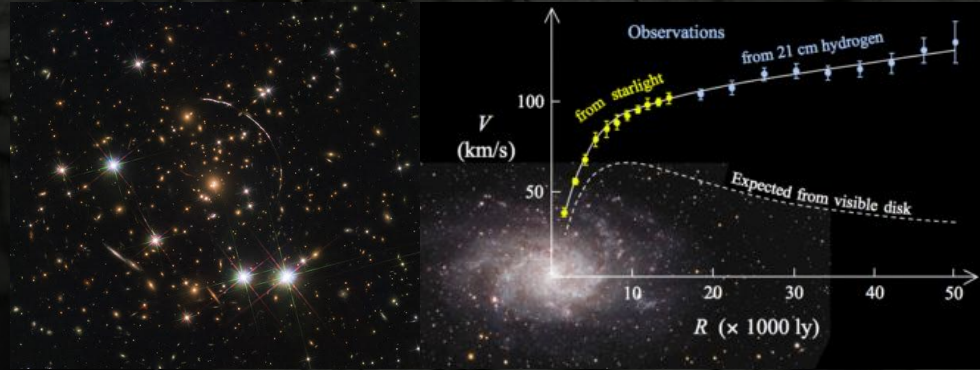
- User authenticates against central production gateway (aCT)
 - so far only one actual user: “production manager” logging on to the aCT machine
 - will expand this to individual LDMX user certificates
- aCT in turn authenticates against ARC instance on sites
- site only accepts jobs from aCT identifying as LDMX production user

Dark matter and LDMX

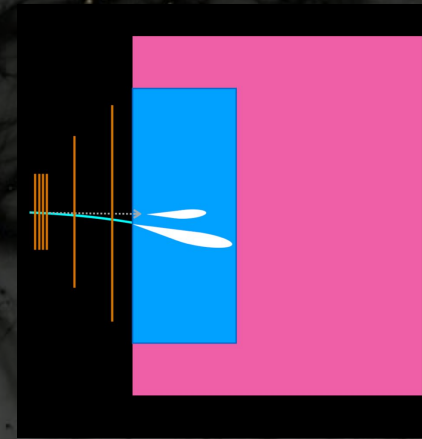
A range of cosmology and astronomical phenomena explained by existence of dark matter

- but so far only gravitational evidence
- no known interaction with visible matter
- → hard to detect it

LDMX aims to produce dark matter in the lab and detect it *through its escape*



typical DM event signature

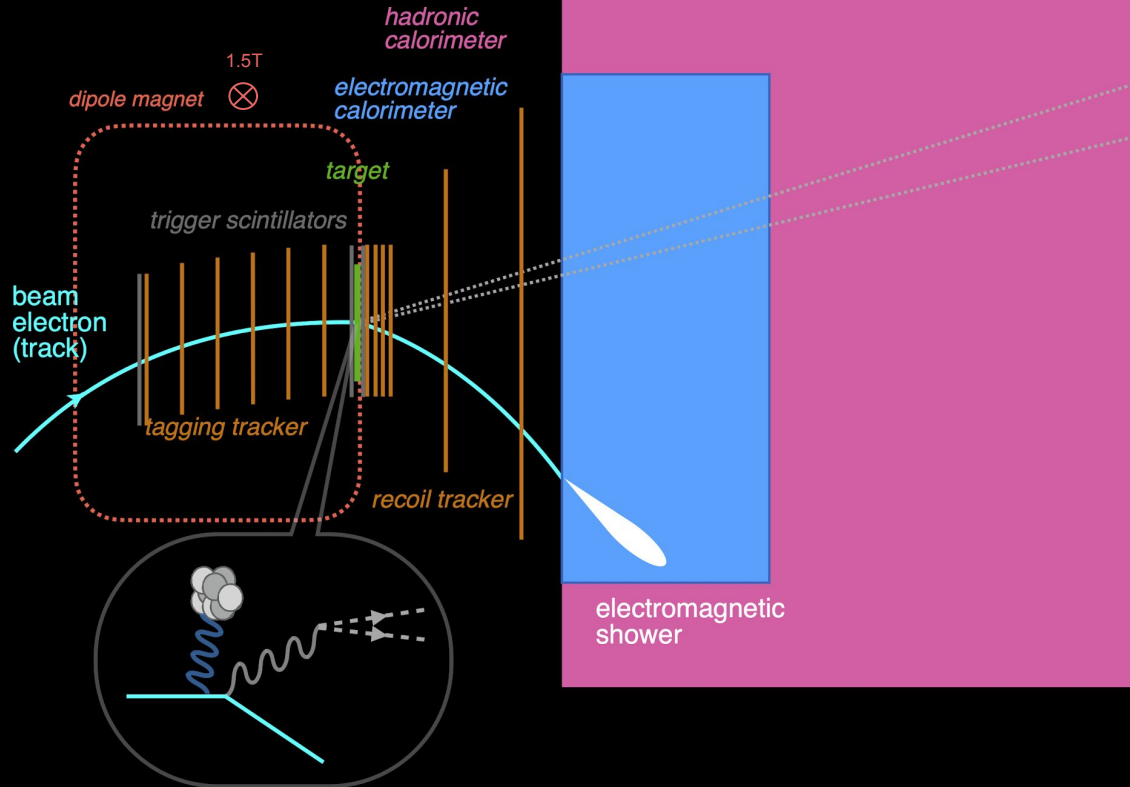


typical background signature

LDMX detector concept

(not to scale)

Signal cartoon

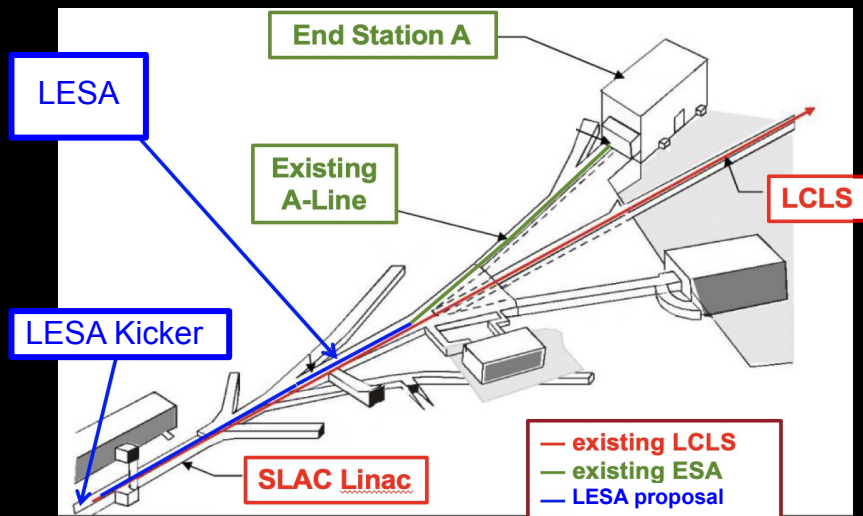


Strategy:

- let electrons hit a target foil
- DM signature: very little activity in the detector
- more likely: photons get radiated → measurable activity

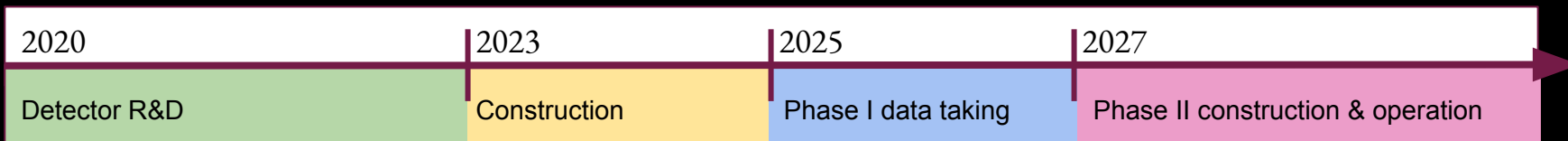
escaping dark matter

LDMX at SLAC



LCLS-2 beam at SLAC:

- main use: electron beam for photon science
 - steal some via Linac to End Station A (LESA)
- 4 GeV beam energy
 - phase-II upgrade at 8 GeV
- low-current
 - measure each incoming and outgoing electron
- fast repetition rate
 - expect 37.2 MHz bucket frequency
 - and $\sim 10^{14}$ electrons on target in 1-2 years



LDMX baseline schedule