Cloud computing for XCMS parameter optimization

wkumler & rlionheart

Goal: compare lots of different XCMS peak-picking parameters to the “gold standard” of manually-integrated peaks.

More general description + background: use a powerful computer to quickly compare performance of automatic, package-defined parameters to user-defined parameters.

`library(xcms)` ← R library

`CentWaveParam()` ← Relevant R function

We have previously fiddled with XCMS settings but never systematically, and they can all have major effects on processing quality.
What I imagine using cloud computing for is iterating over lots of different parameters to see how well they compare to having an undergrad do it by hand.
This would be done by running XCMS a bunch of times with different parameters, then annotating the peaklist and calculating some kind of correlation or quality control metric between the calculated areas and the manual integrations (we probably need to expand on this).

Parameters are all located within the centWaveParam object, from and for XCMS.

Hypothetical code below:

```
library(tidyverse)
library(XCMS)
param_df <- expand_grid(
  ppm=c(1, 2.5, 5, 10, 20),
  peakwidth_min=c(5, 10, 15, 25, 50),
  peakwidth_max=c(5, 10, 15, 25, 50),
  integrate=c(1, 2),
  qscore=c(5, 10, 20, 50),
  extendLengthMSW=c(TRUE, FALSE)
) %>%
  filter(peakwidth_min<=peakwidth_max)
```

Then, run XCMS in some kind of loop, providing different parameters each time:

```
ms_files <- list.files("~/path/to/relevant/mzML/files", pattern =
"mzML")
raw_data <- readMSData(ms_files, mode = "onDisk")
```

```
ingalls_standardss <-
read.csv("http://github/path/to/regina/standards")
manual_values <- read.csv("path/to/manual/integrations.csv")

for(row in seq_len(nrow(param_df))){
  cwp <- CentWaveParam(
    ppm=param_df$ppm[row],
    integrate=param_df$integrate[row],
    qscore=param_df$qscore[row],
    peakwidth=c(param_df$peakwidth_min[row],
                param_df$peakwidth_max[row])
  )
  xdata <- findChromPeaks(raw_data, param = cwp)

 #...other XCMS steps & Will custom code

  xdata_peaks <- chromPeakData(xdata) %>%
    filter(feature_name%in%ingalls_stans$compound_name)
  similarity_vals <- sapply(unique(xdata_peaks$feature_name),
function(j){
    xcms_integrations <- xdata_peaks[xdata_peaks$feature_name==j,]
    manual_integrations <-
manual_values[manual_values$feature_name==j,]
    cor(xcms_integrations, manual_integrations)
  })
}
```

The question is how to best parallelize the process, which is dependent upon the cloud's architecture (?). We can parallelize by file, which is how we've been doing it on laptops and the PLGS:

```
  register(BPPARAM = SnowParam(tasks = length(ms_files)))
```

All the above things or we can do a single file per core, which would probably make best use of our computing hours (assuming each set of params takes about the same amount of time to compute) but I have no idea how this should be done and will depend on resources available to each node (memory, storage, floprate)

```
#!/bin/bash
# Job name:
#SBATCH --job-name=XCMSParams_detailed_params_here
```

```
#
# Account:
#SBATCH --account=whatever
#
# Partition:
#SBATCH --partition=something_from_cloud
#
# Request one node:
#SBATCH --nodes=number_of_param_permutations_to_run
#
# Specify number of tasks for use case:
#SBATCH --ntasks-per-node=number_of_files (maybe?)
#
# Processors per task:
#SBATCH --cpus-per-task=number_of_files (maybe?)
#
# Wall clock limit:
#SBATCH --time=02:00:00
#
# Log file location:
#SBATCH --output="../SLURMlogs/job_%j.out"


## Command(s) to run:
# module load r/3.5.1
# module load r-packages/default
# module load r-packages/custom_ingalls
#
# R CMD BATCH rscripts/XCMS_params_given_params.R
```

Some general questions:

How expensive will this be: price per core-hour? memory per core? What kind of architecture do they use (SLURM? something fancier?)? can we use windows/mac equivalents or do we need to test on a particular platform?

Personal notes from looking through Cloudbank's website:
https://www.cloudbank.org/training

Cost estimates:
The 23 cents rule: Cloud computing breaks down to *roughly* 23 cents/hr for a moderately powerful computer, and *roughly* $23/month for one TB of active data stored. This is for fast access, frequently used data. Archival storage is much cheaper.

CC is pay as you go, so cost management is more important than cost estimate, and all budgets should contain cushions for new questions.

How many instances do you need?
Can we use preemptible machines (probably), and write code so it saves often and helps with ephemeral machines?
Are you stopping and restarting instances of VMs?

Use official cost estimates from vendors in combination with estimates from personal calculations.

Can make a snapshot and replicate 1, 10, 50 identical virtual machines for the computing horsepower, or restart on a small cheap machine for developing and testing code.

Connect to (for example) Azure cloud through the azure portal. It's basically a desktop connected virtually.