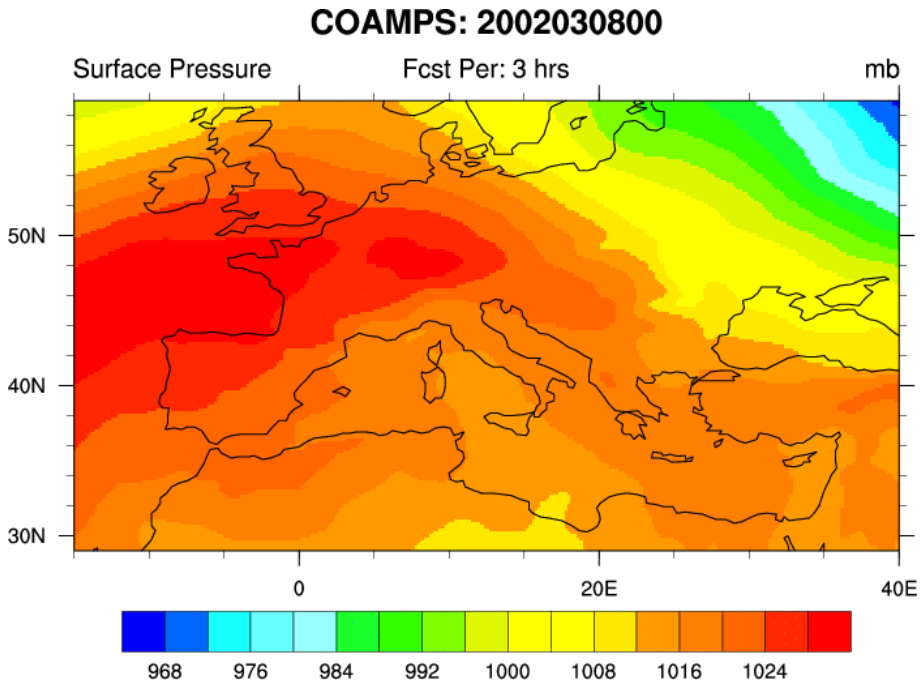
Современные подходы и тенденции обработки научных форматов данных в метеорологии

Котов М.С. (НГТУ, г. Новосибирск)

НАУЧНО-ТЕХНИЧЕСКАЯ КОНФЕРЕНЦИЯ ПО ПРОБЛЕМАМ ГИДРОМЕТЕОРОЛОГИЧЕСКИХ ПРОГНОЗОВ, ЭКОЛОГИИ, КЛИМАТА СИБИРИ (к 40-летию образования СибНИГМИ).19-20 апреля 2011 г. Новосибирск

Этапы

- Обработка данных
- Визуализация данных



Способ хранения

Способ хранения данных во многом определяется способом их получения. Например, файлы, хранящие данные, могут быть текстовыми (матрицы, списки чисел), двоичными (байтовые массивы), представленными в специализированном научном или техническом формате, стандартными графическими файлами.

Популярные научные форматы данных, используемые в метеорологии

- CDF (Common Data Format)
- GRIB (Gridded Binary)
- HDF (Hierarchical Data Format)
- NetCDF

Форматы GRIB и NetCDF

- GRIB – каждая запись предназначена для хранения единственного параметра вместе со значениями , которые представляют собой массив точек или набор спектральных коэффициентов одного уровня, кодированных в виде непрерывного потока битов.
- NetCDF – представляет собой абстракцию, которая описывает данные как коллекцию сомоописываемых объектов, доступных через простой интерфейс. Коллекции поименованных многоразмерных переменных могут быть доступны в произвольном порядке, без знания деталей хранения данных.

Программное обеспечение для работы с форматами GRIB и NetCDF

- Системы инженерных вычислений, такие как MatLab и MathCad
- Специализированные приложения и пакеты, такие как GrADS и NCL
- Утилиты и библиотеки, такие как wgrib, grib_api, g2clib, netcdf

Взаимодействие с форматами из языков программирования

- Из C/C++:
  - Высокая производительность
  - Сложность отладки
- Из интерпретируемых языков:
  - Возможность интерактивной работы

```
[wmsystem@wmsystem ~]$ python
Python 2.7 (r27:82500, Sep 16 2010, 18:03:06)
[GCC 4.5.1 20100907 (Red Hat 4.5.1-3)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>>
>>>
```

  - Более быстрый процесс разработки

Почему Python?

- Интерактивность (в отличии от Perl)
- Наличие такого полезного расширения как NumPy, что позволяет решить проблему производительности для больших структур данных и вычислительных алгоритмов
- Наличие большего количества современных модулей для работы с научными форматами, чем для других высокоуровневых языков программирования
- Легкость написания расширений на C/C++, Fortran (не говоря уже о Pyrex и др.)

Модули для взаимодействия с форматами из Python

- PyNGL/PyNIO – позволяет работать с целым рядом научных форматов
- pygrib – обертка для ECWMF GRIB API C library
- netCDF4 – обертка для netCDF C, C++, and Fortran libraries

Применение в реальной жизни

Задача: разработать систему усвоения данных GRIB и КН-01, обеспечивающую высокую надежность и достаточную производительность

Исходные данные: виртуализированный сервер с ОС Linux (CentOS 5.5 Final), задействовано одно ядро Intel(R) Core(TM)2 Duo CPU E6750 @ 2.66GHz, 512MB оперативной памяти

Рассмотрим подходы к построению системы применительно к данным GRIB

Подход 1

- Использование модуля *struct* для Perl/Python
- Использование связанных таблиц с заранее определенной сеткой в *MySQL/PostgreSQL*

  Результат: очень медленное извлечение данных, медленная вставка, очень медленное обновление, жесткая привязка к сетке. Высокие требования к системным ресурсам и необходимость тонкой настройки системных компонентов.

Подход 2

- Использование модуля *pygrib*, разрабатываемого в ECWMF
- Создание одной записи в БД для одной конкретной точки

  Результат: быстрое извлечение данных, медленная вставка, очень медленное обновление данных. Высокие системные требования к СУБД.

Подход 3

- Предварительная конвертация данных GRIB в NetCDF с помощью *ncl_convert2nc*, и их последующая обработка с помощью модуля *netCDF4*
- Использование массивов (стандарт *SQL99*) в *PostgreSQL* для хранения извлеченных массивов данных

  Результат: более медленная скорость извлечения данных (по сравнению с предыдущим подходом), более низкая надежность из-за дополнительных системных вызовов. Очень быстрая вставка данных и их обновление, гибкость работы с данными.

Подход 4

- Использование модуля *pygrib,* написание на *Pyrex* (смесь C и Python) расширения, куда выносятся ресурсоемкие функции
- Использование массивов (стандарт *SQL99*) в PostgreSQL для хранения извлеченных массивов данных

  Результат: высокие показатели надежности и производительности, при простоте построения системы.

Результаты

Данные приведены для суточного объема телеграмм (~6000)

- Обработка данных из 18000 сообщений GRIB, со вставкой в БД: ~20 минут
- Обработка данных 47000 сообщений КН-01, со вставкой в БД: ~2 минуты

Итого, обработка суточного объема телеграмм с данными GRIB и КН-01, вместе с дополнительными действиями, занимает примерно 25 минут. Данный факт, с лихвой покрывает требование к оперативности, учитывая, что программа обработки данных запускается раз в 3 минуты.

Планы и рекомендации

- Использование модуля *PyNIO* дает небольшой выигрыш в производительности для данных GRIB
- Все тесты проводились с довольно “консервативными” настройками СУБД, более тонкая настройка позволит достичь более впечатлительных результатов
- На “реальном” аппаратном обеспечении, можно достичь 1.5-2х кратного увеличения производительности системы