Fleksible Arbeidsflater for Forskere

Arkitektur for en smidig forskningsinfrastruktur

Kandidatnummer: x

ITLED 4260

UNIVERSITETET I OSLO

[21/12/2018]

Antall ord: 6823

1. Innledning

1.1 Bakgrunn

Denne oppgaven tar utgangspunkt i forskerne som i sin søken etter svar på stadig nye problemstillinger har stadig økende behov for mer regnekraft, smartere utnyttelse av eksisterende regnekraft, nye måter å se på og sammenstille data og nye måter å formidle funn og trene opp nye forskere. Spesifikt ser denne oppgaven på bruk av smidig IT-arkitektur som grunnlag for å kunne håndtere disse behovene.

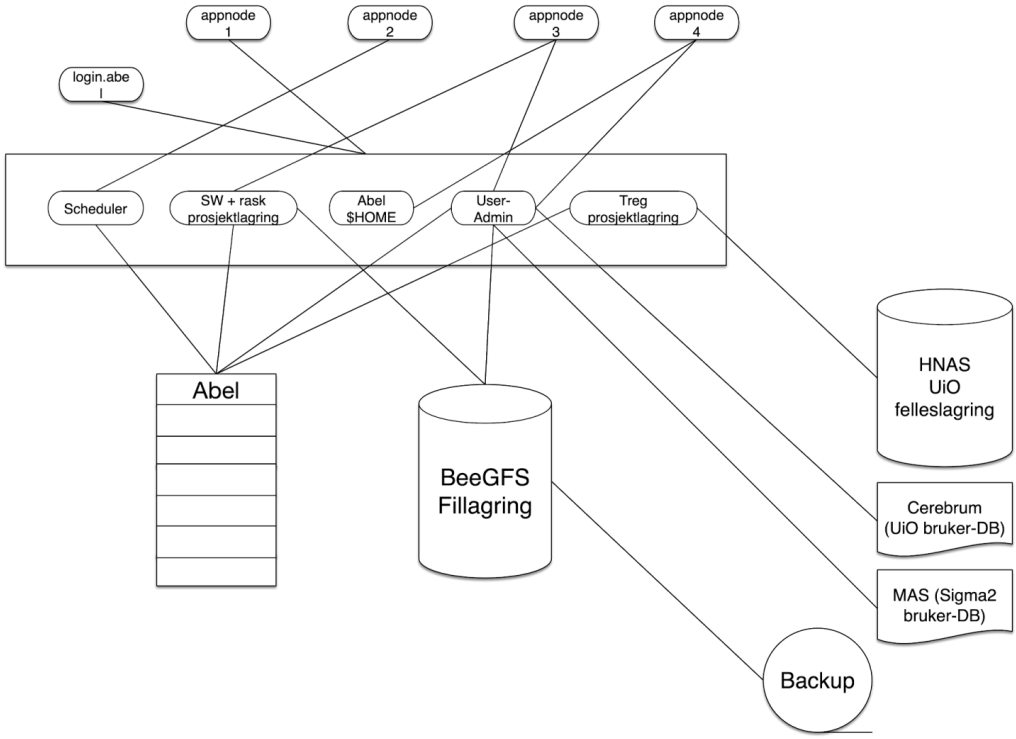
Universitetets Sentrale IT-organisasjon (USIT) sin visjon er forskning og utdanning først. Underavdeling for IT i Forskning bidrar til denne visjonen ved å levere tjenester og infrastruktur til og sammen med forskere, både direkte til UiOs forskere og til Universitets- og Høyskole-sektoren (UH-sektoren) gjennom Metacenter-samarbeidet med Uninett Sigma2 og Norges Arktiske Universitet (UiT), Norges teknisk-naturvitenskapelige Universitet (NTNU), Universitetet i Bergen (UiB) og UiO.

Tradisjonelt har UiT, NTNU, UiB og UiO hatt separate tungregneanlegg (Stallo, Vilje, Hexagon og Abel) med separate driftorganisasjoner. Brukertilgang til disse anleggene var delvis styrt av UNINETT Sigma og Sigma kjøpte også drift-tjenester fra de fire universitetene. Fra 2016 ble det besluttet en ny driftsmodell, der det ble opprettet et nytt selskap til erstatning for Sigma, kalt Sigma2. Dette selskapet fikk ansvar for innkjøp og koordinering av nye tungregneanlegg på vegne av de fire universitetene. Sigma2 skulle også være kontaktpunkt mot brukergruppene som hadde behov for tungregnetjenester. Et førende punkt var også at Sigma2 skulle fortsette å kjøpe drift og brukerstøtte av universitetene for å sørge for at kompetansen ble ivaretatt nærmest mulig forskerne. Et annet førende punkt var at de fire HPC-anleggene ble redusert til to, for på den måten å redusere de totale driftskostnadene. For å sørge for tilgang til nyest mulig teknologi skulle disse to anleggene gis en levetid på fire år, med innkjøp av nytt anlegg hvert annet år. De to første anleggene ble vedtatt plassert ved UiT og NTNU, og det er per nå ingen planer om å ha et større tungregneanlegg plassert ved UiO når Abel fases ut.

I tillegg til å drifte to tungregneanlegg fikk Sigma2 i oppgave å sette opp en data-sentrisk modell der en lagringsressurs og tjenester knyttet til denne skulle settes opp i umiddelbar nærhet av regne-anleggene. Som et svar på denne modellen ble National Infrastructure for Research Data (NIRD) satt opp med et lagringssystem på UiT og et på NTNU og med georeplikering av disse slik at data lagret på det ene systemet skulle bli synlig på det andre systemet.

Selv om tungregneanleggene med tilhørende lagring står for den mest kostbare delen av forskningsinfrastrukturen vi tilbyr, dekker de bare regnebehovene til en begrenset mengde forskningsmiljøer. Tungregneanlegg har blant annet en svært kostbar nettverkstopologi for å håndtere applikasjoner som har behov for svært rask kommunikasjon mellom noder. Dette er essensielt i for eksempel klima-modellering, solfysikk og materialvitenskap, men mindre essensielt innen områder som gensekvensering, maskinlæring og partikkelfysikk der behov for maskiner med mye minne, akselleratorer og mange uavhengige prosessorer er førende. Innen livsvitenskap kommer også behov for håndtering av spesielt sensitive data inn med nye føringer på tungregnearkitekturen.

En kort oppsummering av hva forskningsinfrastruktur på UiO skal dekke er "alt som ikke kan kjøres på forskerens desktop". Vel så viktig som tungregneressurser og spesialtilpasset maskinvare er det å kunne tilby installasjon av og støtte til programvare og gode utviklingsmiljøer. Som en av våre ansatte forklarte for en ledende leverandør av ARM-prosessorer, "de aller fleste av dagens prosessorer blir brukt til å kjøre programvare".



Figur 1: Abel og appnoder sett som plattform-arkitektur. Abel er en CPU-ressurs og BeeGFS er et raskt, distribuert filsystem levert av underavdeling for IT i Forskning. Backup og UiO felleslagring leveres av underavdeling for IT Infrastruktur. Cerebrum leveres av underavdeling for Brukernære Tjenester. MAS leveres av Sigma2. Disse systemene underbygger tjenester for jobb-schedulering, software+rask prosjektlagring, hjemmeområder på Abel, brukeradministrasjon og treg, billig prosjektlagring. Appnoder er fysiske og virtuelle maskiner som benytter en eller flere av disse tjenestene.

Figur 1 illustrer dagens arkitektur for forskningsinfrastruktur på USIT. Abel og det tilhørende lagringssystemet BeeGFS utgjør, sammen med noen støttetjenester, en basis som underbygger fem tjenester:

- En oppgave-schedulerer
- programvare og rask prosjektlagring (delt lagringsområde for et prosjekt)
- Abel hjemmeområde; brukeradministrasjon og
- langsom, billig prosjektlagring.

Disse tjenestene (Abel-plattformen) leveres videre ut til applikasjonsnoder (appnoder), virtuelle eller fysiske servere som står nær Abel med tanke på nettverk. Appnodene kan bruke en, flere eller alle tjenestene i Abel-plattformen, alt etter behovet fra brukergruppen som eier appnoden. Appnodene kan levere beregningstjenester inn til Abel, slik som portaler, grid computing og enkel ssh-tilgang, men benyttes ofte også som separate regneressurser med tilgang til

programvaren på Abel, eller kun for outsourcing av drift og brukeradministrasjon til USIT.

Når det gjelder de virtuelle appnodene er de fleste av disse levert gjennom en VMWare-rigg som leverer VMer som servere driftet av USIT. Vi ser nå at noen forskermiljøer også ønsker tilgang på servere de selv kan drifte, uten delte filområder men med full administrator-tilgang. For disse har USITs alternativ vært å levere tilgang til UH-IaaS, en OpenStack-basert privat sky som opereres i samarbeid mellom USIT og den tilsvarende IT-avdelingen ved UiB. Vi ser for tiden på mulighet for også å levere fullt driftede USIT-servere via OpenStack og på den måten flytte mye av VMWare-bruken over på sky-teknologi.

Selv om modellen i fig. 1 har fungert relativt godt siden Abel kom på plass i 2012 og har gitt oss svært fornøyde brukere, har implementasjonen av den visse svakheter som vi kommer tilbake til i seksjon 4. Den kan hende største utfordringen med dagens arkitektur er dog at Abel ikke blir erstattet av et nytt HPC-system når den skrus av i 2019 slik at arkitekturen uansett må endres.

1.2 Problemstilling og avgrensninger

Tema for denne oppgaven er Agil IT-arkitektur og problemstillingen vi ser på er: *På hvilken måte kan USIT og dens samarbeidspartnere utnytte smidig IT-arkitektur til å levere fleksibel infrastruktur for forskere?*

USIT har en sentral rolle i å levere forskningsinfrastruktur til UiOs forskere. Med forskningsinfrastruktur menes da i hovedsak den maskinvaren som er nødvendig for å dekke forskeres behov for regnekraft og lagringskapasitet og -ytelse, men også mer platform-liknende tjenester som tilgjengeliggjøring av programvare forskerne trenger, håndtering av lisenser for kommersiell programvare dersom den er sentral for forskere.

En viktig avgrensing her er at forskningsinfrastruktur ikke inkluderer oppgaver som mer naturlig løses på forskerens egen arbeidsstasjon eller bærbare PC. Bruk av forskningsinfrastruktur innebærer som sådan alltid at brukeren må få tilgang til og logge seg inn på en tjeneste.

Selv om en signifikant del av vår leveranse av forskningsinfrastruktur går gjennom Metasenter-samarbeidet, vil jeg i denne oppgaven legge hovedvekt på UiOs lokale infrastruktur og forhold når løsning skisseres og heller peke ut muligheter for overføringer til den nasjonale infrastrukturen der det er naturlig.

En interessant komponent i denne problemstillingen ligger i at smidig IT-arkitektur gjerne kobles med smidig utvikling, der smidighet og fleksibilitet ligger i et tjenestelag på toppen av en stabil og alltid tilgjengelig infrastruktur. I forskningsinfrastruktur, der fokus er på å utføre stadig større vitenskapelige problemstillinger på kortest mulig tid, må også den underliggende infrastrukturen være forberedt på stadige endringer.

2. Teoretiske perspektiver

2.1 Hvilke deler av pensum er relevant for problemstillingen?

Problemstillingen, som gitt i avsnitt 1.2, kommer i all hovedsak inn på delen av pensum som omhandler smidig IT-arkitektur. For å kunne besvare problemstillingen må vi (i) forstå hva som ligger i smidig IT-arkitektur og (ii) se på hvilken måte en arkitektur kan bidra til å levere fleksibel infrastruktur.

For forståelsen av smidig IT-arkitektur er (Bloomberg, 2013) mest relevant, men det kan også være interessant å se på Enterprise Architecture og (Ross & al., 2006) for å forstå hva organisasjonens arkitektur bør fokuseres mot. For å se på hvordan en arkitektur kan gi mer fleksibel infrastruktur vil det være relevant for helhetsbildet å se nærmere på tung- og lettvekts IT, plattformer og til en hvis grad skyteknologi.

2.2 Relevant teori

"Agile IT revolution" (Bloomberg, 2013) er skrevet som et slags manifest der forfatteren maler en visjon der dagens Enterprise IT raskt kommer til å bli akterutseilt i en pågående IT-revolusjon identifisert ved fem supertrender:

- *Location Independence* - rent konkret et SOA-prinsipp der den underliggende fysiske implementasjonen av en tjeneste er abstrahert vekk fra selve tjenesten. Trenden er i tillegg fundamental for skyteknologi, men omfatter også mobilitet, der smarttelefoner og tilgjengelighet av nettverk gjør en tilgjengelig uavhengig av fysisk tilstedeværelse.
- *Global Cubicle* - som en forlengelse av lokasjonsuavhengighet kan man sette opp kontoret sitt hvor som helst fra. Sett fra arbeidsgivers side betyr det også at ekspertise kan hentes fra hvor som helst, hvilket bidrar til in- og outsourcing, om arbeidsgiver tillater mangel på fysisk tilstedeværelse.
- *Democratization of Technology* - store enterprise-wide systemer, tunge anbudsprosesser, høy-risiko produksjonssetting er på vei ut, små systemer, fri software, SaaS og abonnementsbaserte tjenester er på vei inn.
- *Deep Interoperability* - teknologier og åpne standarder som web services og REST gir dyp interoperabilitet, men eneste måte å få leverandører til å gå for åpne standarder er å ikke kjøpe fra leverandører som ikke tilbyr det.
- *Complex Systems Engineering* - for å få ekte agilitet må organisasjoner tenke nytt om integrasjoner. Governance tar over for integrasjon, Chief Information Officer blir Chief Governance Officer. EA-rammeverk blir erstattet av best practices for kontinuerlig forretningstransformasjon.

Ny IT-arkitektur må være robust for kontinuerlige endringer. Tjenesteorientert arkitektur (SOA) bygget på riktig implementert Representational State Transfer (REST) og sky-teknologi kommer fullstendig til å erstatte dagens Enterprise IT når syv identifiserte brytningspunkter har manifestert seg:

- Kollaps av Enterprise IT
- tomt for IPv4-adresser; rammeverkenes fall (Zachman, TOGAF og des like)
- Cyberkrig

- Generation Z (nå mer kjent som Millenials) entrer arbeidsplassen
- eksplosjon av Big Data og
- Enterprise applikasjons-kræsj.

Kanskje mest interessant for denne oppgaven er del tre der et forslag på hvordan smidig arkitektur kan implementeres. Om man skjærer gjennom alle forsøkene fra leverandører på å spinne enhver ny teknologi til noe man kan tjene penger på, går tilbake til roten av SOA, REST og Cloud og unngår de vanligste misforståelsene ved REST så kan man implementere ekstremt skalerbare løsninger som vil være i stand til å støtte enhver forretning i kontinuerlig endring.

"Enterprise Architecture As Strategy" (Ross & al, 2006) beskriver et enkelt rammeverk der man, basert på graden av integrasjon og standardisering av forretningsprosesser, kan bygge den arkitekturen som best passer i det landskapet der forretningen opererer. Rammeverket består av en matrise av fire operative modeller:

- Diversifisering - lav prosess-standardisering, lav grad av prosess-integrasjon. Forretningen kjennetegnes av separate enheter med få eller ingen delte kunder og leverandører.
- Replikering - høy prosess-standardisering, lav prosess-integrasjon. Forretningen kjennetegnes av like enheter med få delte kunder. Mange franchise-foretak faller inn under denne kategorien.
- Koordinering - lav standardisering, høy integrasjon. Forretningen kjennetegnes av separate enheter med delt kundemasse og data. Tjenestetilbydere og offentlige institusjoner kommer ofte i denne kategorien.
- Forening - høy standardisering, høy integrasjon. Foreningen kjennetegnes av enhetlige enheter med felles kundemasse, data og prosesser. Flyselskaper er et eksempel hvor forening av enheter er både nyttig og nødvendig.

Når forretningen har funnet ut hvor de er/vil være i matrisen kan den sette opp et kjerne-diagram som beskriver arkitekturen og deretter starte implementering. Som avsjekk på hvor langt forretningen har kommet, defineres det en modenhets-stige på fire trinn:

1) Forretningssiloer
2) standardisert teknologi
3) optimert kjerne
4) forretningsmodularitet.

Med unntak av forretninger i diversifiseringskategorien vil alle ha nytte av å bevege seg opp mot det fjerde trinnet og ideelt vil det ikke lenger være et skille mellom forretning og IT på dette trinnet.

2.3 Empiriske undersøkelser

"ICT Architecture and Project Risk in Inter-Organizational Settings" (Hanseth & Bygstad, 2012) tar for seg to arkitekturer-typer, institutional interface architecture (INA) og service provider architecture (SPA) og ser på risiko knyttet til disse i inter-organisatoriske prosjekter. Empirien var basert på 10 prosjekter i norsk helsesektor over 20 år og viste at det var betydelig

større risiko knyttet til INA enn til SPA. Mens INA innebærer sterke koblinger mellom applikasjonslaget og kommunikasjonslaget og svake koblinger mellom komponentene i kommunikasjonslaget, innebærer SPA sterke koblinger mellom kommunikasjonslagets komponenter, men svake koblinger mellom applikasjonslaget og kommunikasjonslaget. Når de sterke bindingene ligger mellom applikasjonslaget og kommunikasjonslaget og applikasjonene ligger hos de forskjellige organisasjonene blir de tekniske løsningene fort komplekse og koordineringen mellom organisasjonene svært krevende. Til sammenlikning kan en arkitektur med et tett koblet kommunikasjonslag implementeres av et enkelt team og kun et lite stykke utviklingsarbeid kreves av applikasjonsleverandørene.

"Architectural Styles and the Design of Network-based Software Architectures" (Fielding, 2000) oppsummerer Fieldings arbeid fra 1993 til 2000 som arkitekten bak Hypertext Transfer Protocol (HTTP) og Uniform Resource Identifiers (URI), to kjernekomponenter i World Wide Web (WWW), og er som sådan en case-studie av WWW. Avhandlingen definerer Representational State Transfer (REST), arkitekturen bak WWW, og viser hvordan den er bygget opp av forskjellige nettverksbaserte arkitekturer basert på syv arkitektoniske nøkkelegenskaper: ytelse; skalerbarhet; enkelhet; modifiserbarhet; synlighet; portabilitet; og pålitelighet. Ved å se på hvilke begrensninger hver arkitektur legger kan man sette sammen flere arkitekturer og optimere på de nøkkelegenskapene som er ønsket.

En interessant observasjon her er at føringene i REST er lagt på kommunikasjonslaget, alt applikasjonslaget trenger å forholde seg til er en connector. Dette samsvarer godt med observasjonene til (Hanseth & Bygstad, 2012) om service provider architectures.

"The Crossroads of Cloud and HPC: OpenStack for Scientific Research" (OpenStack. 2016) er en case-studie på hvordan OpenStack benyttes i HPC-spesifikke scenarier basert på tilbakemeldinger fra ni organisasjoner. Selv om man kan sette spørsmålstegn ved hvor uhildede case-studier som oppnås når OpenStack ber organisasjoner om tilbakemeldinger om OpenStack, gir studien et interessant overblikk over hvordan OpenStack som skyteknologi kan benyttes for HPC i forhold til virtualisering; nettverks-fabric (intern-nettet på HPC-anlegget); rask, distribuert lagring; last-håndtering; infrastruktur-styring; og federated identity management. Det siste punktet er spesielt interessant for forskningsinfrastruktur med tanke på de nasjonale, nordiske og internasjonale samarbeidene vi er med på innen HPC og eInfrastruktur.

3. Metode

Oppgaven tar utgangspunkt i den kvalitative metode, der datainnsamlingen har bestått i samtaler og dybdeintervjuer med ansatte ved USIT samt dokumentasjon tilgjengelig på USITs nettsider.

Ideelt sett burde man også ha intervjuet et større utvalg av arkitekturens målgrupper, i første omgang forskere og lokale IT-ansvarlige ved UiO. Med tanke på tidsrammen gitt for oppgaven har ikke dette vært gjennomførbart, men for deler av problemstillingen har forfatteren kunnet

støtte seg på en større datainnsamling i regi av USIT rundt behovene knyttet til en sentral Configuration Management Database (CMDB) fra 2017.

Vi har i denne oppgaven foretatt tre dybdeintervjuer; underdirektør for IT infrastruktur (UD-ITF), gruppeleder for Gruppe for brukeropplevelse (GL-UX) samt hovedarkitekten bak Abel og deltaker i den tekniske arbeidsgruppen for innkjøp av Sigma2s HPC-anlegg, ansatt i Gruppe for Forskningsinfrastruktur (ANS-FI). I utvelgelsen av intervjuobjekter har vi lagt vekt på forståelse av forskeres behov sett fra et strategisk, et teknisk og et brukeropplevelsesperspektiv.

Som intervjuform valgte vi å gjøre ustrukturerte dybdeintervjuer. Intervjuene ble tatt opp og varte rundt 60 minutter hver, og deretter gjennomgått og transkribert av undertegnende. De viktigste punktene er beskrevet i seksjon 4.

I forhold til relevante eksisterende løsninger og tjenester har vi også hatt samtaler med gruppeleder for serverdrift og gruppeleder for basis systemdrift bidratt til datainnsamling.

I tillegg til intervjuer og samtaler har USITs prosjekt-nettsider (USIT, 2018) blitt benyttet for å få innblikk i UiOs integrasjonsarkitektur.

4. Funn

Funnene under kommer fra dybdeintervjuer av underdirektør for IT i Forskning (UD-ITF), gruppeleder for Gruppe for brukeropplevelse (GL-UX), ansatt i Gruppe for Forskningsinfrastruktur (ANS-FI), samtaler med gruppeleder Gruppe for Basis Systemdrift (GL-BSD) og gruppeleder Gruppe for Serverdrift (GL-SD), samt USITs interne og eksterne websider (USIT, 2018).

4.1 Strategi for Forskningsinfrastruktur

Som basis for valg av en ny arkitektur for forskningsinfrastruktur må vi vite noe om strategien framover for IT i forskning. Flere interessante temaer om dette kom opp i intervjuet med UD-ITF, i forhold til både nasjonalt og lokalt plan.

4.1.1 Nasjonal strategi

På nasjonalt plan er Sigma2 sin strategi førende, utarbeidet i samarbeid med Metacenteret og bestemt av Sigma2s styre, med representanter fra alle fire universiteter, Uninett, Sintef og en ekstern forsker. Sigma2 søker å maksimere innvirknings- og påvirkningskraft fra vitenskapelig forskning ved å tilby permanent, forutsigbar og kostnadseffektiv eInfrastruktur. Dette skal oppnås ved å

1) tilby avanserte beregnings- og datatjenester som adresserer brukernes behov
2) være fasilitator for tilgang til internasjonale forskningstjenester, skybaserte ressurser og felles komponenter og
3) kombinere Sigma2s infrastruktur med partner-institusjonenes ressurser og kompetanse for å oppnå kostnadseffektive løsninger.

Bak denne strategien ligger det en avklaring av hvilke typer systemer som skal ligge hos hvem, beskrevet ved en matrise med kolonner for finansiering, størrelsesrangering, teknologisk

modenhet, hvem som drifter tjenesten, brukerstøtte-behov, formålsvurdering, tjenestens eier og ansvar for ressursallokering. Ut fra disse faktorene kan man dele systemene i fire kategorier, såkalte tiere, der Tier-0 er internsasjonale HPC-anlegg, Tier-1 er nasjonale HPC-anlegg, Tier-2 er regionale anlegg (f.eks. UH-IaaS som er en OpenStack-installasjon i samarbeid mellom UiB og UiO) og Tier-3 er lokale anlegg på ned på institutt-nivå. Tier-1 skal da opereres av Metacenteret, eies av Sigma2 og tilgang styres av en ressurs-fordelingskomite under Sigma2, mens for Tier-2 er det et løsere koblet samarbeid med vertsinstitusjonene og Sigma2. For Tier-3 er ikke Sigma2 formelt involvert.

Om man følger denne matrisen med tanke på nye teknologier, kan for eksempel innføring av en UH-sky for forskningstjenester begynne på Tier-3 ved at universitetene prøver ut relativt umodne sky-teknologier, bevege seg opp til Tier-2 ved at to universiteter ser at en teknologi er såpass moden at det er verdt å samarbeide om den, og når teknologien er moden og velprøvd nok for permanent og stabil drift så kan den leveres som en Tier-1-tjeneste med felles nasjonal drift[1].

4.1.2 Lokal strategi

Gitt den nasjonale strategien i seksjon 4.1.1 er det åpenbart at UiO selv må dekke endel regnebehov for den lange halen av forskere som har behov for ressurser utover sin egen arbeidsstasjon, men som ikke nødvendigvis trenger et internasjonalt eller nasjonalt HPC-anlegg. Det er også strategisk viktig at universitetene lokalt aktivt opparbeider seg kompetanse og erfaring på nye teknologier for å unngå at teknologien på Tier-1 blir akterutseilt på sikt. Dette gjelder både for ny hardware-teknologi og oppover i stacken på for eksempel sky-teknologi. UD-ITF la derfor vekt på at (1) vi skal fortsette å levere ut appnoder og tilhørende lagring til forskere; (2) disse skal i størst mulig grad leveres via OpenStack; og (3) vi skal fortsette å legge til rette for testing av ny hardware-teknologi.

4.2 OpenStack som en Skyteknologi

OpenStack som skyteknologi var oppe som tema flere ganger i løpet av datainnsamling, både i intervjuene med UD-ITF og ANS-FI og i samtale med GL-SD, med henholdsvis strategisk, driftsteknisk og teknologisk perspektiv.

4.2.1 OpenStack som strategi

Som nevnt i seksjon 4.1.2 er OpenStack viktig som et ledd i å opparbeide kompetanse på ny teknologi. Selv om det nok er lettere og mindre komplekst å fortsette å levere regneressurser på den måten vi har gjort, med manuelt oppsett av hver enkelt appnode og la den kjøre til brukergruppen forsvinner eller den ikke virker lenger, så ligger det lite læring i en slik tilnærming. Ved å tilby OpenStack først ved oppsett av nye appnoder, vil vi måtte lære oss og opparbeide oss erfaring med å tilby ressurser ved hjelp av skyteknologi og brukerne vil opparbeide seg erfaring med å jobbe med en sky-basert ressurs.

[1] Det er her verdt å nevne at denne strategien kom på plass først etter at NIRD tjenesteplattform, basert på den i nasjonal sammenheng lite utprøvde Kubernetes-teknologien, ble installert og vedtatt driftet av Metacenterets driftorganisasjon.

4.2.2 OpenStack som provisjoneringsverktøy

Sett fra et driftsteknisk perspektiv oppleves leveranse av HPC-ressurser via OpenStack svært komplekst. Den kanskje største utfordringen er at mange begreper fra tradisjonell server-drift har fått nye navn i OpenStack, slik at et stort vokabular må læres på nytt uten at innholdet nødvendigvis er veldig ulikt. Det vil si at selv om det er tung kompetanse på tradisjonell server-drift i organisasjonen så må man ta høyde for at en full overgang til OpenStack som provisjonerings-system vil kreve et stort kompetanseløft; kanskje det største kompetanseløftet man har måttet ta i HPC-miljøet siden Beowulf cluster (Becker et al., 1995) introduserte ideen om å sette sammen separate servere i et modulært HPC-cluster i 1994.

4.2.3 OpenStack som skyteknologi

Den kanskje største fordelen med en teknologi som OpenStack er at den gir helt nye muligheter for å tilby stabile tjenester over tid. Siden hardware-laget kan abstraheres bort og brukeren bare ser virtuelle maskiner så trenger ikke lenger brukeren å tenke på utløpte support-avtaler, siden virtuelle maskiner kan migreres fra en server til en annen så trenger ikke brukeren bekymre seg for nedetider og ødelagt hardware. Takket være gode, fleksible APIer, kan brukeren få tilgang til selvbetjeningsløsninger der hun selv kan sette sammen serveren etter egne spesifikasjoner, f.eks. på antall CPUer, minnebehov, lagringsbehov og nettverksbehov. Selve driften av anlegget er altså langt mer frakoblet fra bruken, slik at planlegging av vedlikehold blir enklere og drift for en stor del består i monitorering, automatisering og planlegging av oppgraderinger.

Det bør her også nevnes at OpenStack-installasjonen på UiO er en del av UH-IaaS (UH-IaaS, 2018), og forvaltes i samarbeid mellom UiB og UiO.

4.3 UiO Integrasjonsarkitektur

USIT fikk i 2015 midler til å starte opp Prosjekt UiO Integrasjonsarkitektur der det skulle settes opp en integrasjonsarkitektur for UiO basert på føringene lagt av Difis overordnede IT-arkitekturprinsipper for offentlig sektor (Difi, 2012), UH-sektorens arkitektur (Uninett, 2015) samt UiOs sikkerhetshåndbok (nå Ledelsessystem for informasjonssikkerhet (USIT, 2018).

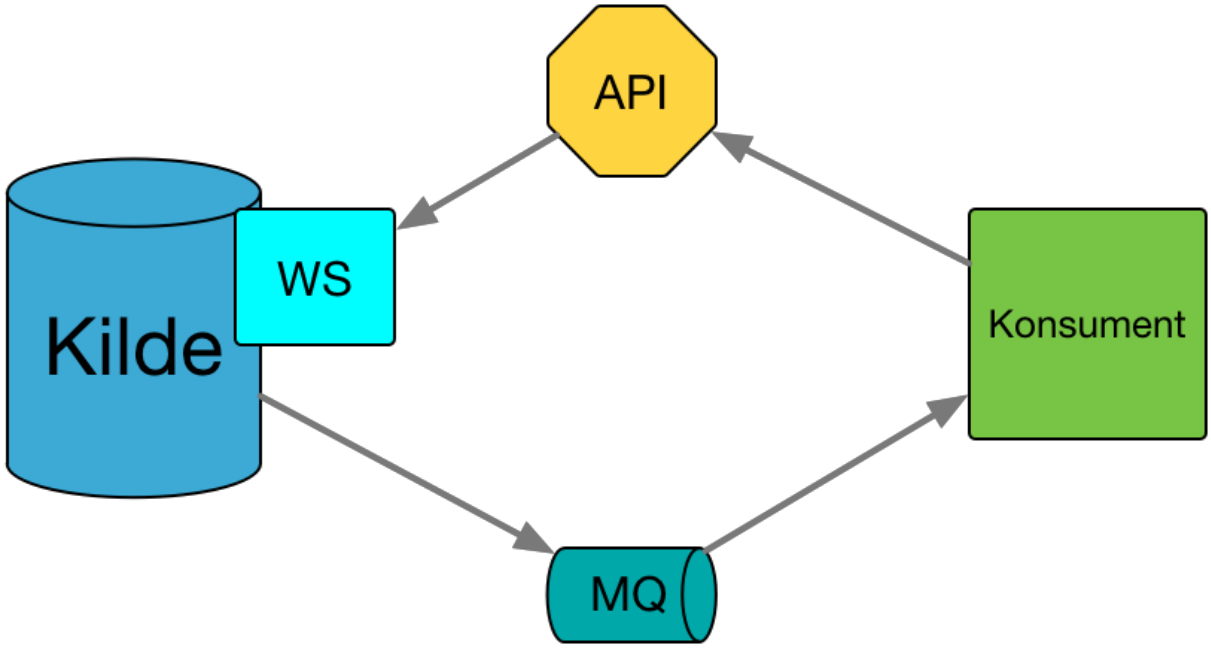
4.3.1 USIT integrasjonsarkitektur - prinsipper

Difis IT-arkitekturprinsipper (Difi, 2018) kan oppsummeres i syv overordnede prinsipper; tjenesteorientering, interoperabilitet, tilgjengelighet, sikkerhet, åpenhet, fleksibilitet og skalerbarhet. Tatt ned på UiO-nivå gir dette at en integrasjonsarkitektur skal være

- Brukerorientert: Brukernes behov endrer seg over tid, derfor må arkitekturen være fleksibel nok til å kunne ivareta behovene også i fremtiden.
- Tjenesteorientert: Integrasjoner skal benytte løse koblinger slik at tjenester og bakenforliggende systemer kan byttes med mindre risiko og kostnader.
- Tilgjengelig: Eier er ansvarlig for at data flere har behov for skal gjøres tilgjengelig for virksomheten via åpne, leverandøruavhengige grensesnitt.
- Oversiktlig: Tjenestene skal registreres i UiOs tjenestekatalog med fullstendige data.
- Etterrettelig: Hvem som har tilgang til data skal registreres.
- Fleksibel: Kost-nytte-vurdering kan benyttes for å begrunne avvik fra prinsippene.

- Effektiv: I de tilfeller der det er effektivt å benytte en tjenestebuss for å knytte tjenester sammen skal tjenestebuss benyttes.
- Veiledet: Et felles kontaktpunkt skal gi veiledning i integrasjonsspørsmål.

4.3.2 UiO Integrasjonsarkitektur og Webservices.



Figur 2: Eksempel på sluttjeneste (konsument) i UiOs integrasjonsarkitektur. Konsumenten får ved hjelp av API manager tilgang til datakilde tilgjengeliggjort via en Web Service (WS). Kilden oppdaterer konsumenten via en meldingskø (MQ).

UiOs integrasjonsarkitektur er basert på Webservices (WS) som teknologi (USIT, 2018). Det er ikke et krav å bruke WS for å følge arkitekturen, men man ser at en del flaskehalser oppstår i tilknytning til oppslag i autorative kilder der det tradisjonelt har blitt brukt en hub-and-spoke modell der datakilden er ansvarlig for å utlevere data til konsumenter. Dette har igjen medført at det er den som sitter med ansvar for kilden som blir sittende med integrasjonsjobben for hver nye konsument. I WS-modellen har systemeier for kilden ansvar for å tilby APIer til konsumenter mens en API-manager håndterer tilgangen til APIene. Med gode APIer kan man da anta at konsumentenes behov for spesialtilpassede integrasjoner vil gå ned og koblingen mellom konsument og kilde løsnes opp. Et annet problem med hub-and-spoke modellen er at integrasjonen foregår ved at hele datasett med alt konsumenten trenger blir lastet ned fra kilden, med de problemer det medfører i forhold til synkronisering, replikering og kontroll av data. I WS-modellen løses dette ved at kilden benytter en meldingskø (MQ) til å gi konsumenter beskjed når data endrer seg. Figur 2 eksemplifiserer hvordan WS kan brukes i integrasjonsarkitekturen. Kilden gir beskjed via MQ at et objekt har endret seg, konsumenten tar kontakt med kildens WS via API manager, henter ned det endrede objektet og oppdaterer på sin side.

4.3.3 UiO integrasjonsarkitektur og REST

Som nevnt i seksjon 4.3.2 er gode APIer en forutsetning for at kilde-eierens integrasjonsarbeidsmengde skal gå ned sammenliknet med arbeidsmengden i hub-and-spoke modellen. Men hva er et godt API? Integrasjonsarkitekten trekker fram REST som metodikk for å oppnå løsere koblinger og deler opp i fire nivåer av REST-API:

- *Nivå 0: RPC over HTTP.* APIer som kaller seg REST men ikke er det. Remote Procedure Call (RPC) kall sendt over nettet, der man typisk implementerer ny funksjonalitet i APIet for hver bestilling. Gir tette koblinger og lite gjenbruk.
- *Nivå 1: Ressurser.* I stedet for spesialiserte kall for hver bestilling, begynner man å samle funksjoner under ressurser, for eksempel alle funksjoner som omhandler navn legges under ressursen *navn*. Kan fortsatt gi spesialiserte funksjoner og lite gjenbruk, men oppfordrer til en modell der ressursen bestemmer mer av strukturen på APIet enn rekkefølgen av bestillinger.
- *Nivå 2: HTTP verb.* Et nivå-2 API er ikke fullt ut REST API, men et godt stykke på vei. Dataene selv representeres som ressurser, HTTP-verbene GET, DELETE, POST og PUT benyttes til å hente, slette, endre og opprette ressurser. Behov for spesialiserte funksjoner blir da minimalt.
- *Nivå 3: Hypermedia Controls/HATEOAS.* For et ekte REST-API skal APIet i WSen være den kontrollerende aktøren i integrasjonen. Her avhenger svaret fra APIet både av tilstanden til dataene og tilgangene konsumenten har. Dette gir navn til begrepet Hypermedia As The Engine Of Application State (HATEOAS), der all nødvendig informasjon om applikasjonens tilstand ligger i hypermedia-dokumentet APIet sender som svar.

Arkitekten påpeker også at å bevege seg fra nivå 2 til nivå 3 er ressurskrevende, slik at det ikke er gitt at nytten er verdt innsatsen.

4.4 Brukerperspektivet

De kanskje mest essensielle behovene som bør dekkes i søken etter en arkitektur for fleksible arbeidsflater for forskere er forskernes behov. Dette var også et viktig tema i intervjuet med GL-UX. En utfordring med å dekke forskernes behov er å finne forskernes behov. Forskere som sådan er en svært kompetent og kunnskapsrik gruppe, men man kan ikke ta for gitt at de er kjent med alle muligheter som ligger i godt interaksjonsdesign. Når man skal lage en løsning, for eksempel for fleksibel forskningsinfrastruktur, bør man alltid starte med brukerbehov og brukeropplevelse. En god metodikk er å gå ut til brukere i målgruppen, finne ut hva de trenger, tegne opp skjermbilder sammen med brukeren for å finne prosessen brukeren trenger (mest) støtte til, og deretter gå tilbake og tegne opp arkitekturen for å finne hvilke komponenter som kan brukes, må endres eller eventuelt implementeres for å støtte prosessen.

Uten direkte å ha snakket med forskere om behov for fleksible arbeidsflater har vi likevel noen punkter som er klare. En observasjon fra GL-UX var at flere av problemene med dagens forskningsinfrastruktur er gjenkjenbare fra arbeidet med MineStudier. Der forskere sliter med å finne fram i en skog av tilgjengelige regnetjenester og -ressurser, sliter studenter med å finne

fram i en skog av tilgjengelige kurs, studie-tjenester og -tilbud. Behovet for en persontilpasset tjeneste som gir oversikt over relevante og aktuelle tjenester vil derfor sannsynligvis være tilsvarende for forskere som for studente. Et godt utgangspunkt er da å samle informasjon og sørge for at den er tilgjengelig via API og deretter bygge tjenesten på toppen ved bruk av interaksjonsdesign.

4.5 Kilder til en fleksibel forskningsinfrastruktur

For å sette opp en fleksibel, oversiktlig forskningsinfrastruktur (og en hvilken som helst annen IT-infrastruktur) trenger man (1) en oversikt over maskinparken (fysiske og virtuelle servere) og (2) styring av identiteter og tilgang. Som påpekt av GL-BSD, dekkes på USIT innsamling av data om datamaskiner av Nivlheim, mens Cerebrum er verktøyet for styring av identiteter og tilgang. Begge benyttes i dag for alle appnoder på lik linje med alle datamaskiner på UiO, begge er utviklet og driftet på USIT og begge følger UiOs integrasjonsarkitektur ved å tilgjengeliggjøre data ved hjelp av API. Man kan derfor anta at nødvendig kompetanse for å lage nødvendige integrasjoner for en fleksibel forskningsinfrastruktur er tilgjengelig på USIT.

5. Diskusjon

Basert på teorien fra seksjon 2 og funnene i seksjon 4 er det tre spørsmål som bør belyses ytterligere før vi kan foreslå en arkitektur for forskningsinfrastruktur. Vi vil i denne seksjonen diskutere:

1) hva ønsker vi at arkitekturen skal løse?
2) hvilke begrensninger bør en smidig arkitektur bør legge?
3) Kan skyteknologi løse forskningsinfrastrukturens utfordringer?

Basert på dette vil vi komme med et forslag til en lokal arkitektur for forskningsinfrastruktur og se på muligheter til å utvide denne til en nasjonal arkitektur for UH-sektoren.

5.1 Hva skal arkitekturen løse?

Før en setter opp en IT-arkitektur bør man ha en idé om problemet, altså hva arkitekturen skal løse. Det antakelig største problemet med dagens løsning for forskningsinfrastruktur er mangelen på oversikt, både for brukere og administratorer. Dagens løsning med appnoder kan ses på som et eksempel på organisk tilblivelse av arkitektur. Konseptet med appnoder startet smått som en god ide for raskt å hjelpe enkelte forskningsmiljøer som hadde behov utover det et tungregneanlegg kunne løse. Appnoder utgjør nå 30 virtuelle og fysiske servere der brukere logger direkte inn, 99 servere om man teller alle med tilknytting til Abel. Det er ingen sentral oversikt over informasjon om support-avtaler, kontrakter og utfasingsplan og det er ingen samlet oversikt over hvilke appnoder som bruker hvilke tjenester fra Abel. Den beste oversikten ligger i en manuelt vedlikeholdt HTML-tabell på internsidene.

For brukerne er det ingen sentral oversikt over tilgjengelige appnoder, annet enn noen felles regneressurser som har vært heldige og kommet inn på dokumentasjonssidene. Tilgang til og kjennskap til appnoder eid av prosjekter og grupperinger på UiO har for en stor del vært overlatt til eierne.

Den valgte arkitekturen bør altså understøtte at brukere får (1) oversikt over tilgjengelige ressurser og (2) oversikt over de ressurser de har tilgang til. Brukere omfatter her både de som skal benytte ressursene og de som skal administrere og forvalte ressursene, slik at informasjon som gis om ressursene bør differensieres i forhold til brukerens rolle.

5.2 Hvilke begrensninger skal arkitekturen ha?

Om man ser på matrisen for plassering av en organisasjon i forhold til operativ modell i (Ross et al., 2006) kan UiO plasseres i kategorien for koordinering, med lav grad av standardisering og høy grad av integrasjon av prosessene. Samtidig er UiOs integrasjonsarkitektur basert på REST, som innebærer en høy grad av standardisering. I agile arkitekturer for DevOps (Bloomberg, 2013) abstraheres hardwaren vekk under et virtuelt lag for å sørge for en stabil, standardisert platform man kan utvikle smidige tjenester oppå. I en tungvekts/lettvekts IT-terminologi (Bygstad, 2017) kan man til en hvis grad se på tungvekts IT som platformen for å kunne tilby lettvekts IT. Spørsmålet ser altså ut til å være hvor i arkitekturen man skal legge standardiseringen; i prosessene, i plattformen, eller i transportlaget.

Om man ser på forskjellen mellom institusjonell interface arkitektur (INA) og tjenesteleverandør arkitektur (SPA) beskrevet av (Hanseth & Bygstad, 2012) ligger hovedforskjellen i arkitekturen i hvilke koblinger som er løse og hvilke som er sterke. Der de sterke koblingene ligger mellom applikasjonslaget og kommunikasjonslaget i et inter-organisatorisk prosjekt (INA) er risikoen langt høyere enn i prosjekter der applikasjonslaget og kommunikasjonslaget er løst koblet og de sterke koblingene ligger i et standardisert kommunikasjonslag (SPA). Dette stemmer også godt for WWW (Fielding, 2000) der kommunikasjonslaget er sterkt standardisert og klientenes kommunikasjon med ressurser er abstrahert vekk fra det fysiske laget gjennom HTTP[2].

En måte å se det på er at alle arkitekturer har en eller flere bindinger eller begrensninger (constraints (Fielding, 2000)) som legger føringer på systemet. Om man legger begrensingen i transportlaget og ikke i interaksjonslaget blir valget av arkitektur langt mindre avhengig av brukerfunksjonalitet og arkitekturen blir mindre sårbar for endringer i brukernes ønsker og dermed mer smidig. Som en heldig bieffekt for forskningsinfrastruktur legger et sterkt standardisert transportlag heller ikke føringer på hardware-laget, slik at behovet for agilitet og eksperimentering på server-nivå også kan opprettholdes.

5.3 Kan skyteknologi løse alt?

Om man tar (Bloomberg, 2013) helt bokstavelig vil kun løsninger basert på SOA, REST og Cloud stå igjen når IT-revolusjonen er over, alle de syv brytningspunktene har manifistert seg og støvet har lagt seg. Samtidig henviser han også til det Agile Manifest og dets første bud om at individer og interaksjoner ligger over prosesser og verktøy.

Når det gjelder skyteknologi på UiO og andre steder i UH-sektoren er det OpenStack (OpenStack, 2016) som har fått mest fotfeste. Som det kom fram i seksjon 4.2 ses det på som

[2] Man kan også si at WWW må anses som et inter-organisatorisk prosjekt.

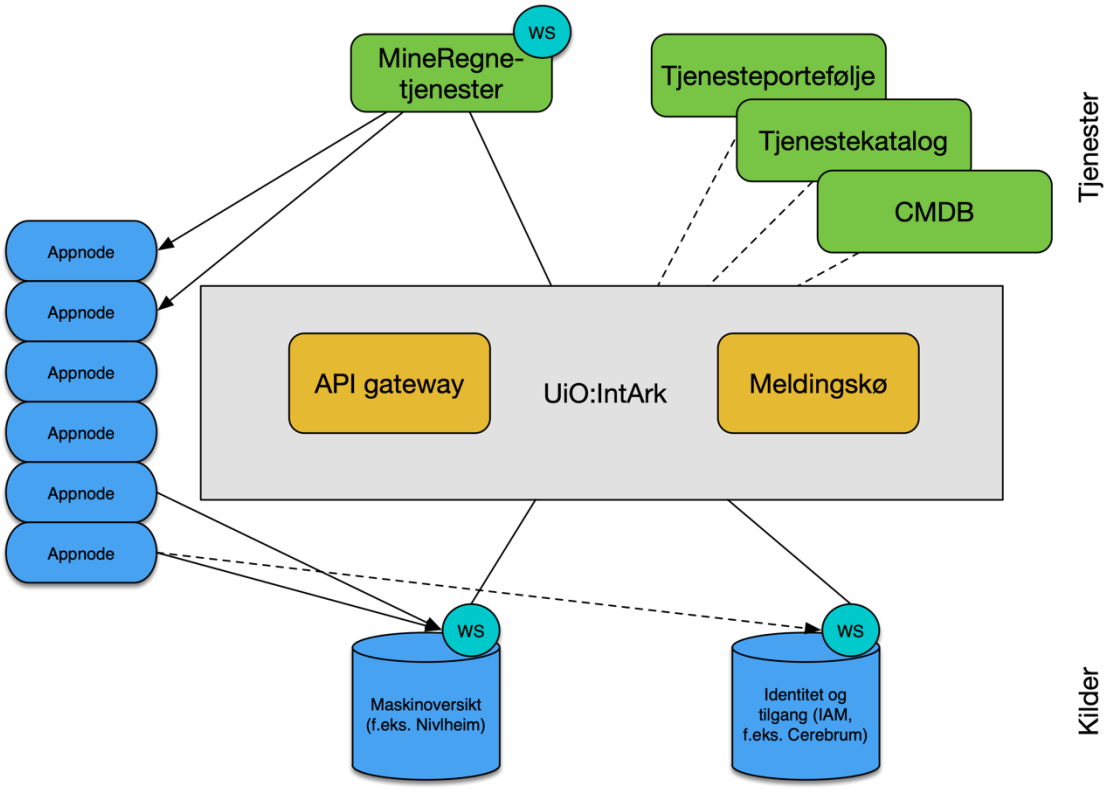
strategisk viktig å levere mest mulig av regneressursene via OpenStack. Men kan man se for seg at alle regneressurser skal tilbys fra OpenStack?

HPC er essensielt forskjellig fra skyteknologi. HPC-anlegg er svært begrensede i størrelse sett i forhold til hva de store skytilbyderne kan tilby. Sist HPC-verden hadde en større revolusjon var da (Becker et al., 1995) introduserte Beowulf-clusteret i 1994. Dette var på et tidspunkt da WWW fortsatt var i startgropen, 6 år før (Fielding, 2000) introduserte REST, løse bindinger var et ukjent tema og det nærmeste man hadde APIer var RPC. Målet med et HPC-cluster er å knytte CPUer tettest mulig sammen, med lavest mulig latens på nettverket og størrelsen på anlegget avhenger av midlene man har til rådighet ved innkjøpstidspunkt. Målet med et sky-anlegg er å gi elastisk tilgang til løst koblede ressurser skalert etter behov, og størrelsen på anlegget avhenger av hvor mye kunder er villige til å betale for.

Fordelene med skyteknologi kommer best til syne når størrelsen på systemet øker. For spesialisert hardware i små mengder blir det fort komplekst og dyrt. Om man ser på casene i (OpenStack, 2016) så er det relativt tydelig at det kreves spesialtilpasninger for å oppnå HPC-ytelse. Det synes klart at det er strategisk lurt å tilby appnoder i OpenStack, særlig der behovet som skal dekkes er generisk, uten særlige krav til ytelse eller hardware-arkitektur. For enkelt-noder eller få noder for testing av nye hardware-arkitekturer, HPC-anlegg og spesialtilpassede FPGA-maskiner vil det fortsatt være behov for løsninger utenfor OpenStack, i alle fall fram til neste revolusjon.

5.4 Forslag til lokal arkitektur

Ut fra Difis, UH-sektorens og universitetets arkitektur-prinsipper må arkitekturen for forskningsinfrastruktur være tjenesteorientert, interoperabel, tilgjengelig, sikker, åpen, fleksibel og skalerbar. Det er ingen indikasjoner, verken i teori eller funn i denne oppgaven, på at en smidig arkitektur ikke bør følge disse prinsippene.



Figur 3: Arkitektur for fleksible forskningsflater. MineRegnetjenester får tilgang til APIene til kildene for Maskinoversikt (Nivlheim) og Identitet og tilgang (Cerebrum) via UiO IntArk og får oversikt over hvilke appnoder som er tilgjengelige og hvilke appnoder innlogget bruker har tilgang til. Endringer i tilganger og maskintilstand varsles gjennom meldingskø.

Et forslag til en slik arkitektur er gitt i Figur 3. Arkitekturen baserer seg på UiO IntArks lette tjenestebuss, altså API gatewayen og meldingskøen. To kilder, Maskinoversikt og AIM presenterer sine APIer via API Gateway og pusher endringer via meldingskøen. MineRegnetjenester får tilgang til APIene via gatewayen og henter nødvendig informasjon til å generere oversikt over hvilke appnoder som er tilgjengelige og hvilke appnoder som innlogget bruker har tilgang til. All maskinspesifikk informasjon (ressursens tilstand) ligger på hver appnode, og pushes til Maskinoversikt. Maskinspesifikk informasjon inkluderer informasjon om hvilke(t) filsystemer som er tilgjengelige på appnoden, installert programvare, eier, administrator, dokumentasjon om avvik fra standard installasjonsopplegg og så videre. All kommunikasjon er REST-basert og MineRegnetjenester presenterer selv et fullverdig REST API, der HATEOS sørger for at det som presenteres er tilpasset innlogget brukers rolle og tilganger. Eksempler på roller er normal bruker, administrator, eier eller prosjektadministrator.

Et viktig aspekt i forhold til fleksibilitet og skalerbarhet er at det er hver enkelt appnode som er autorativ kilde for egen tilstand, Maskinoversikt skal kun samle inn informasjon. En utfordring her kan være grenseoppgangen mellom appnoder og IAM. Slik det er satt opp i dagens system ligger noe tilgangsinformasjon på appnodene (brukere og grupper som kan logge inn) mens noe maskininformasjon ligger i Cerebrum (f.eks. IP- og MAC-adresser, kontaktinformasjon).

Som eksempler på fleksibiliteten i arkitekturen viser de stiplede boksene i Figur 3 forslag til andre mulige tjenester som kan gjenbruke oppsettet uten større endringer i APIer;

tjenestekatalog kan genereres med et tilleggsfelt i maskininformasjon om hvilke tjenester som kjøres på maskinen (og alternativt hvilke andre maskiner som kjører tjenesten), tjenesteportefølje trenger i tillegg informasjon om start- og sluttdato for tjenesten. En fremtidig felles Configuration Management Database (CMDB) (USIT, 2018) kan tenkes som en relativt enkel utvidelse av MineRegnetjenester til å inkludere all maskinvare som er koblet til UiOs nett, med et mulig behov for en tilleggskilde for fysisk lokasjon av enhetene.

Merk også at det ikke er noe i arkitekturen som utelukker verken skyteknologi som OpenStack eller HPC-anlegg. Det OpenStack i bunn og grunn tilbyr er virtuelle maskiner, og disse kan sende informasjon til Nivlheim på samme måte som andre virtuelle maskiner. For HPC-anlegg er det alltid en eller flere innloginsnoder, og det er jo nettopp disse som ga ideen til appnoder i første omgang.

Et interessant moment i figur 3, om man trekker inn en av de fem supertrendene fra (Bloomberg, 2013) om demokratisering av teknologi, er at begge de to kildene, Nivlheim og Cerebrum, er systemer utviklet lokalt på USIT, mens API-manageren og meldingskøen begge er opensoure (hhv. Gravitee.io og RabbitMQ). Aktiv bruk av Nivlheim som kilde til informasjon om appnoder vil sannsynligvis medføre behov for nye data, hvilket kan antas å være mer komplisert i et enterprise-system fra en større leverandør enn i en egenutviklet lettvektsløsning.

5.5 Utvidelse til arkitektur for UH-sektoren

Forskningsinfrastruktur er antakelig det området som har kommet lengst på sammarbeid i UH-sektoren, med Uninett Sigma2 og Metacenteret med UiT, NTNU, UiB og UiO som sammen eier og drifter nasjonale HPC-anlegg, samarbeider tett om avansert brukerstøtte og deler kompetanse innen HPC. I tillegg ser vi at

- UiO og UiB samarbeider om OpenStack gjennom UH-IaaS
- IT-BOTT (IT-samarbeidet mellom universitetene i Bergen, Oslo, Trondheim og Tromsø) har klare intensjoner om å bruke UiO:IntArk som utgangspunkt for BOTT:IntArk
- UiO har tatt til ordet for å få på plass UH:IntArk og
- Uninett er i gang med bestilling av en felles IAM for UH-sektoren (Uninett, 2018).

Det er derfor naturlig å vurdere eventuell utvidelse av arkitekturen i figur 3 til å inkludere hele UH-sektoren. Om vi antar at UH:IntArk og UH:IAM kommer på plass er det tre alternativer som synes mulige (illustrert i henholdsvis figur 4a, 4b og 4c):

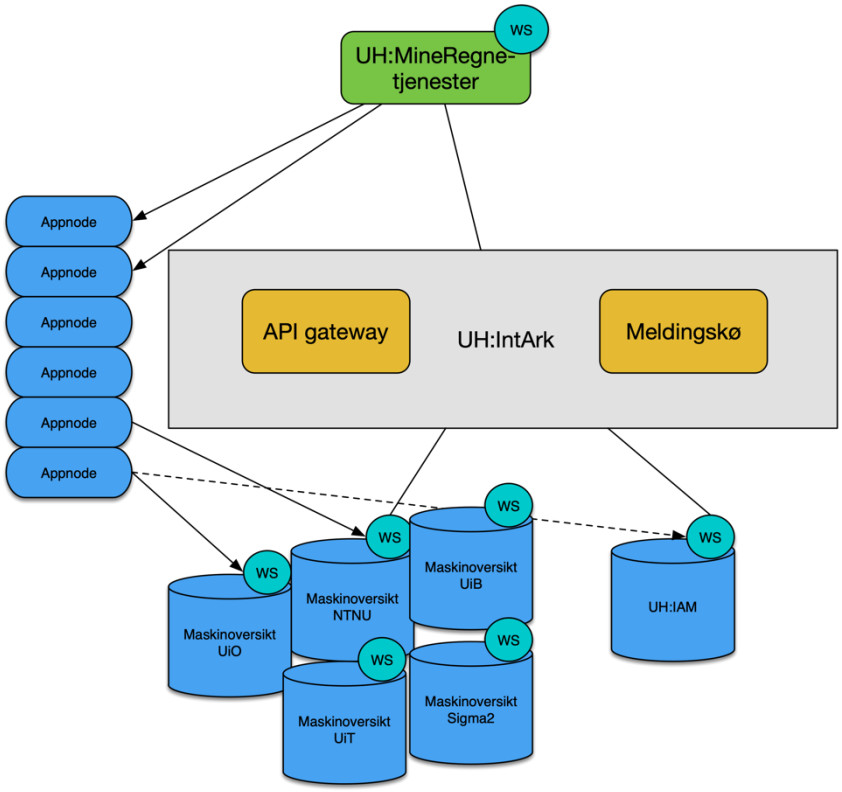
1. Erstatte UiO:IntArk med UH:IntArk og erstatte IAM med UH:IAM. Legge til en kilde for maskinoversikt for hvert universitet samt en for Sigma2. Lage en felles MineForskningstjenester. Utfordringen vil da ligge i å sørge for at APIene for hver maskinoversikt fungerer likt. Man kan også oppleve det som tyngre å få på plass eventuelle endringer i en felles UH:IAM enn i en lokal IAM ettersom flere interessenter må involveres.
2. Replikere hele arkitekturen for hvert universitet samt for Sigma2. Benytte UH:IntArk til å koble sammen MineForskningstjenester på hvert universitet. Dette krever at hver site har eller lager maskinoversikt og IAM som tilbyr API.

3. Hvert universitet samt Sigma2 tilbyr egne MineForskningstjenster som hver tilbyr tilsvarende API, koblet sammen gjennom UH:IntArk. Dette krever minst standardisering, men forskjeller i underliggende arkitekturer kan gjøre det mer komplekst å tilby samme API.

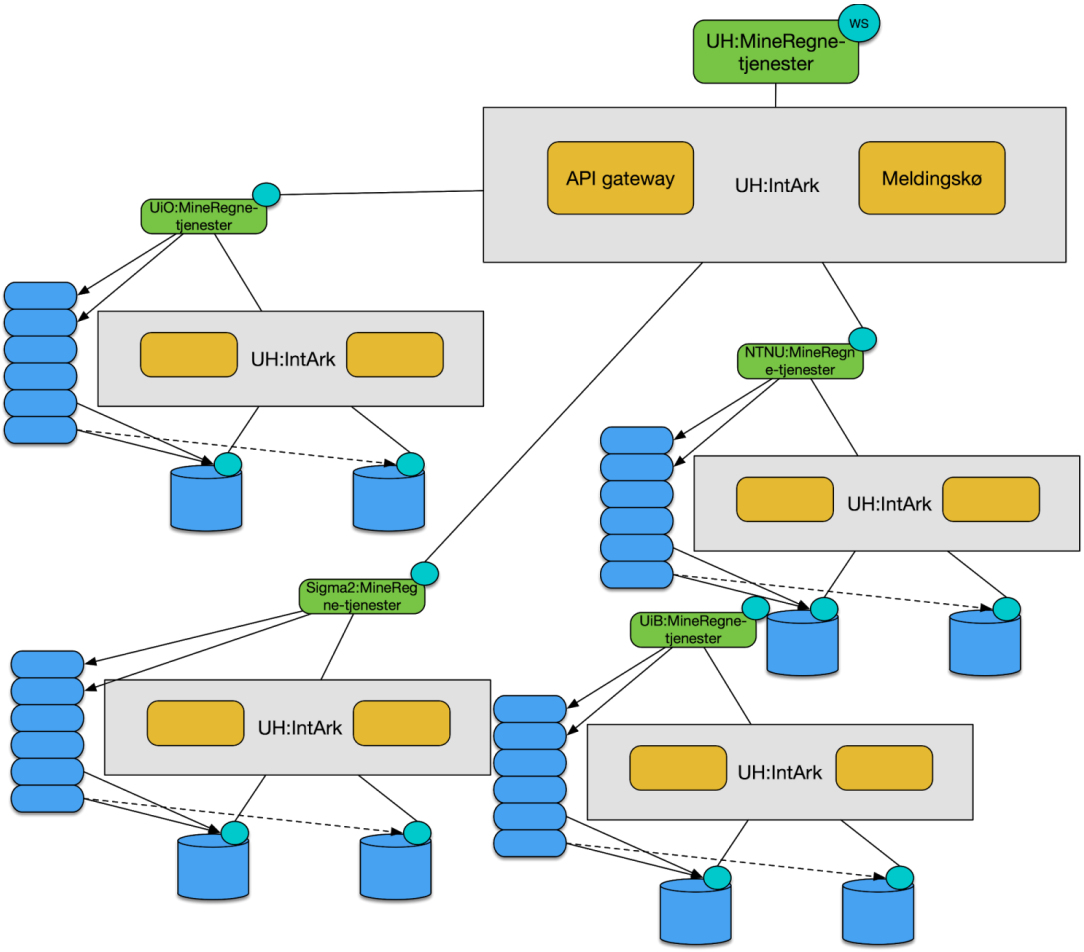
Om vi går tilbake til de operative modellene i (Ross & al, 2006) kan vi si at alternativ 1 kan settes i kvadranten for høy prosess-standardisering og høy integrasjon (forening); alternativ 2 gir høy standardisering og lav integrasjon (replikering); mens alternativ 3 gir lav standardisering (hver institusjon har sin egen arkitektur) men høy integrasjon (felles integrasjonsarkitektur) (koordinering).

Om vi går tilbake til arkitekturen for WWW (side 84, Fielding, 2000) er REST en lagdelt arkitektur der klienten ikke ser noen av de underliggende arkitekturene. Dette samsvarer godt med alternativ 3.

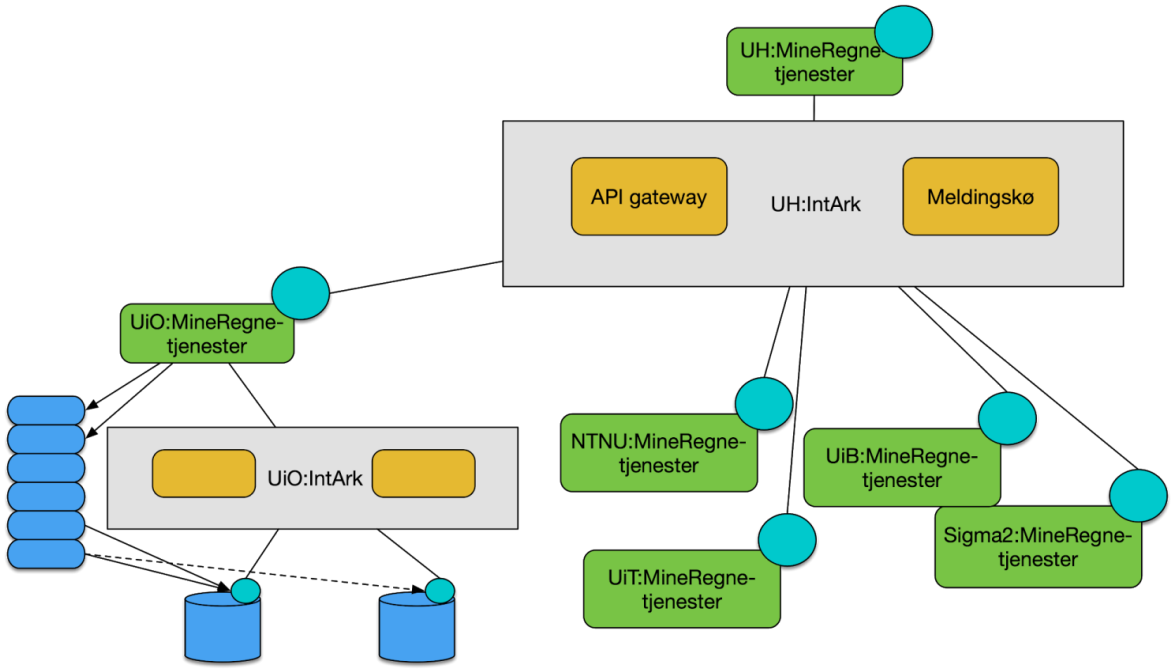
Her kan vi altså si at alternativ 3 gir godt samsvar med både Enterprise Arkitektur og med mer moderne smidig arkitektur.



Figur 4a: UH:MineRegnetjester med site-spesifikke maskinoversikt-tjenester. Høy grad av standardisering og integrasjon.



Figur 4b: Arkitekturen er replikert på hver site. Høy grad av standardisering, lav grad av integrasjon.



Figur 4c: UH:MineRegnetjenester integrerer med site:MineRegnetjenester. Ingen sentral føring på underliggende arkitektur. Høy grad av integrasjon, lav grad av standardisering.

6. Konklusjon: Et sammendrag for beslutningstakere, med handlingsalternativer og anbefaling

Målet med arkitektur generelt er å skape funksjonelle rom for mennesker. Målet med IT-arkitektur er å skape funksjonelle IT-systemer for mennesker. Når det gjelder forskningsinfrastruktur er den funksjonell når mennesker, både brukere og administratorer kan få oversikt over ressurser som tilbys, tilgang til eksisterende ressurser og mulighet til å prøve ut nye typer ressurser mens de fortsatt er nye. Først når brukere og administratorer har nok oversikt og arkitekturen er fleksibel nok til å holde tritt med de stadige endringene for IT i forskning kan man si at arkitekturen bak forskningsinfrastrukturen er funksjonell nok.

Problemstillingen som gitt i seksjon 1.2 var: På hvilken måte kan USIT og dens samarbeidspartnere utnytte smidig IT-arkitektur til å levere fleksibel infrastruktur for forskere?

Om vi går tilbake til Fig. 1 er Abel og BeeGFS ressursen og appnodene benytter tjenester basert på denne ressursen. Nå som Abel er på vei til å skrus av er tiden kommet for å løfte plattform-laget et hakk høyere og se på appnodene som ressursene vi forvalter og la enkel, oversiktlig tilgang til ressursene og god støtte i effektiv utnyttelse av ressursene være tjenestene vi tilbyr.

En arkitektur som skissert i figur 3, bygget på REST og web services, med base i UiO:IntArk og bruk av Nivlheim for maskinoversikt og Cerebrum (eller den IAMen som blir valgt i UH-sektoren) som kilder bør være et godt utgangspunkt for en funksjonell arkitektur for UiOs forskningsinfrastruktur.

Merk at den skisserte løsningen er basert på SOA og REST som originalt beskrevet, ikke på leverandørers og konsulenters forståelse av disse teknologiene. Det er derfor lite sannsynlig at en tilfredsstillende løsning kan kjøpes på markedet. Man kan også si at oversiktlig tilgang til maskinvare er en kjerneverdi for forskningsinfrastruktur, slik at egenutvikling av MineRegnetjenester er en god bruk av ressurser.

Vi anbefaler at midler settes av til et prosjekt for å utvikle MineRegnetjenester ved UiO med utgangspunkt i arkitekturen i figur 3. Her kan vi dra erfaring fra MREG sommerprosjekt (USIT, 2018) der et system for DNS-håndtering ble implementert i løpet av sommeren 2018 ved hjelp av IFI-studenter kombinert med tilstedeværelse av folk med rett kompetanse i USIT. Med tanke på behovet for HATEOAS i MineRegnetjenester kan det være lurt å rekruttere studenter som ennå ikke har lært at REST er vanskelig.

Med tanke på utvidelse av arkitekturen til hele UH-sektoren anbefaler vi å arbeide for alternativ 3 i seksjon 5.5, altså at hvert universitet bygger sin versjon av MineRegnetjenester og at disse bindes sammen ved hjelp av UH:IntArk. En slik arkitektur er godt fundamentert i litteraturen for smidig arkitektur samt i Difis arkitekturprinsipper og bør passe godt til UH-sektorens organisatoriske utforming. En relativt fungerende versjon satt i produksjon på UiO bør være på plass før et slikt arbeid starter for å sikre godt momentum.

7. Referanseliste

Bloomberg, J. (2013). The agile architecture revolution: how cloud computing, rest-based SOA, and mobile computing are changing enterprise IT. John Wiley & Sons.

Jeanne W. Ross, Peter Weill, David Robertson (2006). Enterprise Architecture As Strategy: Creating a Foundation for Business Execution. Harvard Business School Press.

Hanseth, Ole and Bygstad, Bendik, (2012). ICT ARCHITECTURE AND PROJECT RISK IN INTER-ORGANIZATIONAL SETTINGS. ECIS 2012 Proceedings. 130.

Fielding, Roy Thomas (2000). Architectural Styles and the Design of Network-Based Software Architectures. Ph.D. Dissertation. University of California, Irvine. AAI9980887.

OpenStack Foundation (2016). The Crossroads of Cloud and HPC: OpenStack for Scientific Research: Exploring OpenStack Cloud Computing for Scientific Workloads. CreateSpace Independent Publishing Platform. ISBN 1539673987, 9781539673989.

UH-IaaS (2018) “A Norwegian Cloud Infrastructure for Research and Education”. http://www.uh-iaas.no/

USIT (Desember, 2018) "Prosjekter ved USIT - Universitetets senter for informasjonsteknologi.". https://www.usit.uio.no/prosjekter/.

Becker, Donald J., et al. (1995) "BEOWULF: A parallel workstation for scientific computation." Proceedings, International Conference on Parallel Processing. Vol. 95.

Difi (2012) "Overordnede IT-arkitekturprinsipper for offentlig sektor - Difi." https://www.difi.no/sites/difino/files/arkitekturprinsipper-2.1_0.pdf

Difi (2018) "Arkitekturprinsipper | Difi." . https://www.difi.no/fagomrader-og-tjenester/digitalisering-og-samordning/nasjonal-arkitektur/prinsipper.

UNINETT (2015) “Felles IKT-arkitekturprinsipper for universitets- og høgskolesektoren”, https://www.uninett.no/sites/default/files/webfm/arkitekturprinsipper%20UH%20V18.pdf

USIT (2018) “Ledelsessystem for informasjonssikkerhet”, https://www.uio.no/tjenester/it/sikkerhet/lsis/index.html.

USIT (2018) “Webservice – hva og hvordan”, https://www.uio.no/tjenester/it/sikkerhet/integrasjonsarkitektur/hjelp/webservice.html

USIT (2018) “Configuration Management DataBases - CMDB”, https://www.usit.uio.no/prosjekter/cmdb/

Bygstad, B. (2017) “Generative innovation: a comparison of lightweight and heavyweight IT”. J Inf Technol (2017) 32: 180, https://doi.org/10.1057/jit.2016.15

Uninett (2018) “Identity and Access Management - IAM”, https://www.uninett.no/iam

USIT (2018) “MREG sommerprosjekt - sluttrapport”, https://www.usit.uio.no/om/organisasjon/iti/gd/aktiviteter/dnsinfo-som-selvstendig-losning/mreg-sommerprosjekt-sluttrapport.html