黑客派

Why we picked AKKA cluster as our micr oservice framework

```
<p>申明：  本文是从国外技术网站上看到的。觉得比较好久摘录下来了。仅做学习之用。<br> <a hr
f="https://link.hacpai.com/forward?goto=https%3A%2F%2Ftech.iheart.com%2Fwhy-we-pick
d-akka-cluster-as-our-microservice-framework-bbf3019a3217%23.jdf1r8wwu" target="_blank
 rel="nofollow ugc">点击此处查看原文链接，需要翻墙</a></p>
<script async src="https://pagead2.googlesyndication.com/pagead/js/adsbygoogle.js"></scr
pt>
<!-- 黑客派PC帖子内嵌-展示 -->
<ins class="adsbygoogle" style="display:block" data-ad-client="ca-pub-5357405790190342"
data-ad-slot="8316640078" data-ad-format="auto" data-full-width-responsive="true"></in
>
<script>
    (adsbygoogle = window.adsbygoogle || []).push({});
</script>
<blockquote>
 <p>Recently at iHeartRadio we decided to migrate our one monolithic Java backend service
nto multiple microservices, more specifically we decided to implement these microservices as
Akka apps and use Akka cluster to weave them into our microservice cluster. This post is abou
 how we reached this decision. There are three main sections in this post: first a brief discussi
n of the goals we wanted to achieve with microservices, second, the specific reasons why we t
ink Akka apps without Http interface makes the most sense for our goals, and third, a brief lo
k at the initial architecture design.</p>
</blockquote>
<h2 id="Goals-we-want-to-achieve">Goals we want to achieve</h2>
<h2 id="Goal-1--Easier-and-faster-release-cycle-">Goal 1, Easier and faster release cycle.</h
>
<p>Monolithic code base for the Java backend is one of the major factors preventing us from
being able to release services at a more granular pace. Code changes to multiple services hav
 to be tested/QA as a whole, which means small changes have to wait for all other changes (re
evant or irrelevant), before they can be released. We want to address this problem with micro
ervices that can be released on a microservice by microservice basis—each microservice can
ave its own release schedule. Thus we can deliver new features/fixes to clients at a faster pace
with smaller steps).</p>
<h2 id="Goal-2--Improve-development-productivity-with-looser-and-clearer-inter-module-
ependencies">Goal 2, Improve development productivity with looser and clearer inter-modul
 dependencies</h2>
<script async src="https://pagead2.googlesyndication.com/pagead/js/adsbygoogle.js"></scr
pt>
<!-- 黑客派PC帖子内嵌-展示 -->
<ins class="adsbygoogle" style="display:block" data-ad-client="ca-pub-5357405790190342"
data-ad-slot="8316640078" data-ad-format="auto" data-full-width-responsive="true"></in
>
<script>
    (adsbygoogle = window.adsbygoogle || []).push({});
</script>
<p>In our monolithic Java backend code, different functional areas depend on each other tig
tly. The dependencies are also hard to track without carefully inspecting the code, thus makin
 them harder to manage. These over-dependencies make the whole code cumbersome to ch
nge—to change code at one place you may have to change code at multiple places accordin
ly. The implication of such changes are hard to understand. By dividing code into clearly sepa
ated microservices, the dependencies are much looser through messages and easier to inspec
 in simple configuration files.</p>
<h2 id="Goal-3--Better-reusability-composability">Goal 3, Better reusability/composability<
h2>
<p>Classes in a monolithic backend tend to grow larger and larger congregating logic from d
```

```html
fferent functional areas, which makes reusing more and more difficult. We want to take this o
portunity to redesign the modules so that each microservice will have a smaller interface and
learly defined responsibilities. This will make it easier to reuse them as modules and compose
higher level microservices with lower level ones.</p>
<h2 id="Goal-4--Easier-team-integration">Goal 4, Easier team integration</h2>
<p>The monolithic backend codebase is huge in size and very complex to understand. It crea
es a higher barrier for developers from outside the dedicated backend team to contribute to
he backend. Code size within each microservice, on the other hand, is much more modest and
easier to learn. This opens the doors to different development organizations, such as having cl
ent developers contributing directly to the code base or a more vertically oriented team struc
ure.</p>
<script async src="https://pagead2.googlesyndication.com/pagead/js/adsbygoogle.js"></scr
pt>
<!-- 黑客派PC帖子内嵌-展示 -->
<ins class="adsbygoogle" style="display:block" data-ad-client="ca-pub-5357405790190342"
data-ad-slot="8316640078" data-ad-format="auto" data-full-width-responsive="true"></in
>
<script>
    (adsbygoogle = window.adsbygoogle || []).push({});
</script>
<h2 id="Why-we-picked-Akka-cluster-as-the-core-architecture-for-our-microservice">Why
e picked Akka cluster as the core architecture for our microservice</h2>
<p>Now let me go through a few reasons why we made this pick:</p>
<blockquote>
 <ul>
  <li>Out-of-the-box clustering infrastructure</li>
  <li>Loose coupling without the cost of JSON parsing.</li>
  <li>Transparent programming model within and across microservices.</li>
  <li>Strong community and commercial support</li>
  <li>High resiliency, performance and scalability.</li>
 </ul>
</blockquote>
<h3 id="Out-of-the-box-clustering-infrastructure">Out-of-the-box clustering infrastructure<
h3>
<p>One of the costs of microservices is the clustering infrastructure you need to build—that
ncludes but is not limited to discovery, load balancing, monitoring, failover and partitioning t
e microservices. There are 3rd party tools that can help with these clustering functionalities, b
t they require a strenuous integration effort and introduce significant complexity to the stack.
Akka cluster provides these clustering infrastructure components out of the box. We had the c
uster up and running with only a couple lines of configuration changes. Actually, these cluster
ng functionalities are so ready that Typesafe implemented their general distributed system m
nagement tool Conductr using Akka cluster.</p>
<h3 id="Loose-relatively--coupling-without-the-cost-of-JSON-parsing">Loose(relatively) co
pling without the cost of JSON parsing</h3>
<p>One of the common protocols used by microservices is HTTP with JSON body. This setup
has a performance cost of text (JSON) parsing and a development cost of writing JSON parser
. Akka’s message communication over binary protocol is more optimized for performance a
d there is no extra parsing code to write. We value both benefits over the looser coupling pro
ided by HTTP with JSON. We also separate our message API with the service implementation
o provide enough decoupling between the clients and services.</p>
<script async src="https://pagead2.googlesyndication.com/pagead/js/adsbygoogle.js"></scr
pt>
<!-- 黑客派PC帖子内嵌-展示 -->
<ins class="adsbygoogle" style="display:block" data-ad-client="ca-pub-5357405790190342"
```

```
data-ad-slot="8316640078" data-ad-format="auto" data-full-width-responsive="true"></in
>
<script>
    (adsbygoogle = window.adsbygoogle || []).push({});
</script>
<h3 id="Transparent-programming-model-within-and-across-microservices">Transparent pr
gramming model within and across microservices</h3>
<p>The Akka programming model, actor model, is transparent within and across microservic
s. All the calling is done through asynchronous message passing regardless of whether the cal
er and responder are on the same microservice or different ones. This single programming m
del within and across microservices has two major benefits:1) it makes development experien
e consistent—when writing clients, you don’t need to remember where the service actor is,
ocal or remote. 2) it makes it easier to move logic around, you can easily merge or split micro
ervices with little-to-no code change.</p>
<h3 id="High-resiliency--performance-and-scalability">High resiliency, performance and scal
bility</h3>
<p>Apache Spark uses Akka under the hood for driver-worker communication. Their team is
uilding masterless standalone Spark mode using Akka cluster. Not much more was needed to
convince us that it will comfortably satisfy our performance requirements.</p>
<h3 id="Strong-community-and-commercial-support">Strong community and commercial s
pport</h3>
<p>The Akka community is unquestionably substantial and vigorous, but what sets Akka diff
rent from other candidates in our list is the option of having commercial support from Typesa
e, which is provided by the core developers in the Akka teams. It’s precious for us to have th
 confidence that you will always be able to get the “most correct” answer to related proble
s and you never get blocked by a technical problem for more than 24 hours.<br> With these
eatures of an Akka cluster tallying well with our goals, it was an easy decision to make.</p>
<script async src="https://pagead2.googlesyndication.com/pagead/js/adsbygoogle.js"></scr
pt>
<!-- 黑客派PC帖子内嵌-展示 -->
<ins class="adsbygoogle" style="display:block" data-ad-client="ca-pub-5357405790190342"
data-ad-slot="8316640078" data-ad-format="auto" data-full-width-responsive="true"></in
>
<script>
    (adsbygoogle = window.adsbygoogle || []).push({});
</script>
<h2 id="Microservices-with-Akka-Cluster">Microservices with Akka Cluster</h2>
<p>As a final note, here is a brief review our initial architecture design of the microservice pla
form.</p>
<p><img src="https://static.hacpai.com/images/img-loading.svg" alt="akka" data-src="https
//cdn-images-1.medium.com/max/800/0*XRpHE0KYm4nAo9SW.png"></p>
<p>All microservices are implemented in Akka apps and running in a single Akka cluster. Ab
ve this Akka cluster layer, a REST layer is composed of one or more Play! Web applications ser
ing as Http (mostly JSON) public interfaces for the microservices. They communicate with the
microservices by having router actors deployed inside the akka cluster—we call these actors
gents. The Web apps also handle some cross cutting functionalities such as security and cach
ng. Below the Akka cluster is the lower data storage layer, which represent the root sources of
data/information for our backend.<br> Instances of microservices can join and leave Akka clu
ter according to demand. When a redundant instance of a microservice joins the cluster, all m
mbers get notified, and will automatically start to include that instance when load balancing r
quests to the service. Same is true when an instance leaves the cluster, members will get notif
ed that it’s leaving the cluster. This way we can scale up and down at a service by service lev
l.<br> There you go. As of now we have several microservices live in production and the clust
r has served us very well. I will post further updates, hopefully good ones, later.</p>
```

原文链接：Why we picked AKKA cluster as our microservice framework

```
<script async src="https://pagead2.googlesyndication.com/pagead/js/adsbygoogle.js"></scr
pt>
<!-- 黑客派PC帖子内嵌-展示 -->
<ins class="adsbygoogle" style="display:block" data-ad-client="ca-pub-5357405790190342"
data-ad-slot="8316640078" data-ad-format="auto" data-full-width-responsive="true"></in
>
<script>
    (adsbygoogle = window.adsbygoogle || []).push({});
</script>
```